

통계계산소프트웨어

# 데이터 셋의 결합 및 관리

2018. 10.

# 데이터 셋 결합의 종류

- 세로결합

- ✓ 두 데이터 셋을 수직적으로 결합하는 것을 ‘데이터 셋의 연결 (concatenation)’이라고 한다. ‘덧붙인다’로 표현

◎ 변수기준으로 합친다

- 가로결합

- ✓ 두 데이터 셋을 수평적으로 결합하는 것으로 ‘데이터 셋의 통합(merge)’이라고 한다. ‘병합한다’로 표현

◎ 관측기준으로 합친다

# SAS Data Set의 세로 결합 예

NAME	MATH	STAT
김철수	90	34
강민호	67	78

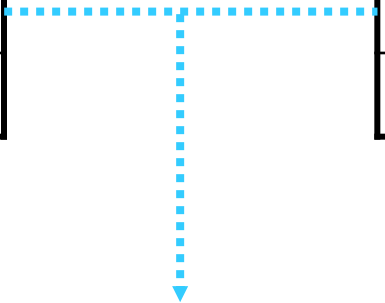
NAME	MATH	STAT
이영희	98	100
박지수	34	98

NAME	MATH	STAT
김철수	90	34
강민호	67	78
이영희	98	100
박지수	34	98

# SAS Data Set의 가로 결합 예

NAME	MATH
김철수	90
강민호	67

NAME	STAT
김철수	100
강민호	98



NAME	MATH	STAT
김철수	90	100
강민호	67	98

# 1. 세로결합: SET 명령문

✓ 기존의 생성된 Data Set을 불러 오는데 이용한다.

```
DATA new_data_set ;  
  SET old_data_set;
```

## Example

```
DATA test1;  
  INPUT x1 @@;  
CARDS  
4 2 3 4 5  
;  
  
DATA test2;  
  SET test1;  
  x2 = 3;  
  x3 = x1 + x2;  
RUN;
```

## 출력 결과

X1	x2	x3
4	3	7
2	3	5
3	3	6
4	3	7
5	3	8

# SET 문장의 다양한 사용

## ■ 복사(COPY)

- ✓ 데이터 셋 전체를 복사할 경우 (old → new), 즉 새로운 데이터 셋 new가 기존의 데이터 셋 old와 똑같이 생성,  
이름만 다른 데이터 셋(new)이 하나 더 생기는 셈

```
DATA new;  
SET old;
```

## ■ 변수의 부분선택(subsetting)

- ✓ 기존의 데이터 셋 old에 있는 몇 개의 변수를 제외하고 old를 복사하여 새로운 데이터 셋 new를 만들 때 사용

```
DATA new;  
SET old;  
DROP variables;
```

## ■ 연결 (concatenation)

- ✓ 기존에 구성되어 있는 두 개 이상의 SAS 데이터 셋을 순서대로 연결 하여 새로운 SAS 데이터 셋을 생성

DATA sample3;

SET sample1 sample2;

- ✓ sample3에는 먼저 sample1의 내용이 있고 그 아래 sample2의 내용이 붙어 있음
- ✓ concatenate 결합을 할 때, SET 문장 내의 data set은 그들이 나열된 순서대로 순차적으로 읽혀짐
- ✓ 새로운 data set은 모든 input data set으로 부터 모든 변수와 총 관측치들을 모두 포함

## 예7.2 SET 명령문을 이용한 세로결합

```
DATA male;  
  INPUT name $ sex $ mid final pre;  
CARDS;  
김철수 M 10 40 30  
강민호 M 50 15 45  
; RUN;
```

VIEWTABLE: Work.Male					
	name	sex	mid	final	pre
1	김철수	M	10	40	30
2	강민호	M	50	15	45

```
DATA female;  
  INPUT name $ sex $ mid final ;  
CARDS;  
이영희 F 15 10  
박지수 F 20 .  
; RUN;
```

VIEWTABLE: Work.Female				
	name	sex	mid	final
1	이영희	F	15	10
2	박지수	F	20	.

```
DATA combine;  
  SET male female;  
  IF final=. THEN final=mid;  
RUN;
```

VIEWTABLE: Work.Combine					
	name	sex	mid	final	pre
1	김철수	M	10	40	30
2	강민호	M	50	15	45
3	이영희	F	15	10	.
4	박지수	F	20	20	.



## 2. 가로결합: MERGE 명령문

- ✓ SAS Data Set의 가로 결합에 사용된다.
- ✓ 동일한 변수 이름에 같은 자료가 있는 경우 나중에 언급된 자료로 대체된다.
- ✓ 이 경우 만약 뒤의 값이 결측값이면 결측값으로 대체된다. 이 단점을 보완한 것이 **UPDATE명령문**이다. UPDATE명령문의 기능은 MERGE명령문과 같으며, 단지 나중의 Data Set에 결측값이 있을 경우 MERGE문과는 달리 기존의 자료값을 유지한다.
- ✓ 만약 특정 변수에 대하여 MERGE 혹은 UPDATE 명령을 수행하는 경우 **BY 명령문**을 사용할 수 있으며, **이 경우 사용되는 모든 DATA Set이 PROC SORT를 이용하여 사전에 정렬되어야 한다.**

```
DATA new_data_set ;  
    MERGE(UPDATE) data_set_1 data_set_2 ...;  
    BY var1 var2 ...;
```

# MERGE 명령문 활용 예

```
DATA combine;  
  MERGE data1 data2;  
RUN;
```

NAME	MATH
김철수	90
강민호	67

data1

NAME	MATH	STAT
김철수	89	100
강민호	.	98

data2

NAME	MATH	STAT
김철수	89	100
강민호	.	98

combine

# UPDATE 명령문 활용 예

```
DATA combine;  
  UPDATE data1 data2;  
RUN;
```

NAME	MATH
김철수	90
강민호	67

data1

NAME	MATH	STAT
김철수	89	100
강민호	.	98

data2

NAME	MATH	STAT
김철수	89	100
강민호	67	98

combine

# Example; MERGE, UPDATE 명령문(예 7.4)

```
DATA mid;
  INPUT name $ sex $ pre mid ;
CARDS;
김철수 M 30 10
강민호 M 45 50
이영희 F . 15
박지수 F . 20
최병호 F . 8
;RUN;
```

```
DATA final;
  INPUT name $ sex $ pre final;
CARDS;
이영희 F 32 10
김철수 M . 40
박지수 F 15 20
강민호 M . 15
;
RUN;
```

```
PROC SORT DATA=mid; BY name; RUN;
PROC SORT DATA=final; BY name; RUN;
```

```
DATA all;
  MERGE mid final;
  BY name;
RUN;
DATA all2;
  UPDATE mid final;
  BY name;
RUN;
```

	name	sex	pre	mid	final
1	강민호	M	.	50	15
2	김철수	M	.	10	40
3	박지수	F	15	20	20
4	이영희	F	32	15	10
5	최병호	F	.	8	.

	name	sex	pre	mid	final
1	강민호	M	45	50	15
2	김철수	M	30	10	40
3	박지수	F	15	20	20
4	이영희	F	32	15	10
5	최병호	F	.	8	.

# 데이터 셋의 가로 결합-새로운 변수 추가

## □ PROC SORT

데이터셋을 지정한 변수로 정렬하는데 이용한다. SAS의 Data Step문에서 자주 사용하는 명령어 중 하나이다.

PROC SORT 는 항상 **by문과 함께 사용**된다.

숫자는 missing, 음수, 영, 양수의 순으로 정렬하며  
문자는 아래의 순서로 정렬한다.

빈칸 ! # \$ % & \* + - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @  
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]  
abcdefghijklmnopqrstuvwxyz{/~

★ **내림차순으로 정렬하고 싶은 경우는 descending 옵션을 사용**

```
proc sort data= a;  
by descending a;  
run;
```

# 데이터 셋의 가로 결합-새로운 변수 추가

- ✓ 기존의 데이터 셋에 새로운 데이터 셋을 가로결합 하는데 있어서 변수만을 추가하고자 할 때 Merge 명령문과 IF 명령문을 함께 사용하여 해결한다.

- **Example: (예제)**

```
DATA mid;  
  INPUT name $ sex $ pre mid ;  
CARDS;  
김철수 M 30 10  
강민호 M 45 50  
이영희 F . 15  
박지수 F . 20  
최병호 F . 8  
; RUN;  
DATA final;  
  INPUT name $ sex $ pre final;  
CARDS;  
이영희 F 32 10  
김철수 M . 40  
박지수 F 15 20  
강민호 M . 15  
이만호 F 50 100  
; RUN;  
PROC SORT DATA=mid; BY name; RUN;  
PROC SORT DATA=final; BY name; RUN;  
DATA all3; MERGE mid final; BY name; IF mid; RUN;  
DATA all4; MERGE mid final; BY name; IF final; RUN;
```



VIEWTABLE: Work.Mid

	name	sex	pre	mid
1	강민호	M	45	50
2	김철수	M	30	10
3	박지수	F	.	20
4	이영희	F	.	15
5	최병호	F	.	8



VIEWTABLE: Work.Final

	name	sex	pre	final
1	강민호	M	.	15
2	김철수	M	.	40
3	박지수	F	15	20
4	이만호	F	50	100
5	이영희	F	32	10



VIEWTABLE: Work.All3

	name	sex	pre	mid	final
1	강민호	M	.	50	15
2	김철수	M	.	10	40
3	박지수	F	15	20	20
4	이영희	F	32	15	10
5	최병호	F	.	8	.



VIEWTABLE: Work.All4

	name	sex	pre	mid	final
1	강민호	M	.	50	15
2	김철수	M	.	10	40
3	박지수	F	15	20	20
4	이만호	F	50	.	100
5	이영희	F	32	15	10

### 3. 데이터 셋 옵션의 사용

✓ 해당 Data Set 이름 뒤에 ( )안에 넣어 사용한다.

#### KEEP option

Data Set에 포함될 변수들을 지정한다.

**KEEP= *variable* [ *variable* ... ]**

#### DROP option

Data Set에 제외할 변수들을 지정한다.

**DROP= *variable* [ *variable* ... ]**

#### RENAME option

Data Set에 있는 기존 변수의 이름을 바꾼다.

**RENAME=( *Old\_Name*=*New\_Name*[*Old\_Name*=*New\_Name*])**



## Example : 데이터 셋 옵션(예7.6)

```
DATA all;  
  MERGE mid(KEEP=name mid pre OBS=4)  
        final(DROP=sex RENAME=(pre=pre1));  
  BY name;  
RUN;  
PROC PRINT DATA=all;  
RUN;
```

```
DATA all1;  
  MERGE mid final;  
  BY name;  
  RENAME mid=m_score final=f_score;  
  DROP sex pre;  
RUN;  
PROC PRINT DATA=all1(FIRSTOBS=2 OBS=3);  
TITLE '2-nd and 3-rd observation';  
RUN;
```

OBS	name	pre	mid	pre1	final
1	강민호	45	50	.	15
2	김철수	30	10	.	40
3	박지수	.	20	15	20
4	이만호	.	.	50	100
5	이영희	.	15	32	10

*2-nd and 3-rd observation*

OBS	name	m_score	f_score
2	김철수	10	40
3	박지수	20	20

## 4. DO-END 명령문

- ✓ DO와 이에 짝을 이루는 END명령문 사이에 있는 명령문을 반복 수행하도록 한다.

```
DO variable = start TO end BY increment;
```

- ✓ start는 시작 숫자를 지정하고, end는 마지막 숫자를 넣는다.
- ✓ BY 문 뒤에는 각 단계에서의 증분을 넣는다.
- ✓ BY 문 뒤를 생략하면 증분은 1이 된다.
- ✓ 수행 과정에서의 결과를 Data Set에 저장하기 위해서는 **OUTPUT 명령문을 내부에 사용**하여야 한다.
- ✓ 다음과 같은 형식의 사용도 가능하다.

```
DO variable = number_1, ..., number_n;
```

```
DO variable = 'character_1', ..., 'character_n';
```

## 4. DO-END 명령문

```
data ab;
  wage = 100 ;
  do i = 1 to 12 ;
    earned + wage ;
  end; run;
proc print; run;
```

[illegible]

# 단순 반복

```
DO index-variable = start TO stop BY increment;  
  SAS statements;  
END;  
Start=시작, stop=끝, increment=증가분(default=1)
```

예) 1년 동안 1000원을 투자하여 연이율 7.5%일 때, 이자수입 계산

DATA a;

amount=1000;

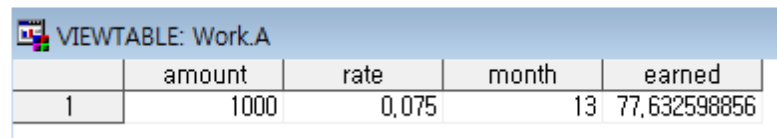
rate=0.075;earned=0;

DO month=1 to 12;

earned+(amount+earned)\*(rate/12);

END;

RUN;



	amount	rate	month	earned
1	1000	0.075	13	77.632598856

- DO루프의 12번째 수행이후 month의 값은 13으로 증가, 13은 지정된 정지값을 넘어서기 때문에 수행을 멈춤

# OUTPUT 명령문

- ✓ DO명령문 내에서 수행된 결과를 Data Set에 저장하기 위하여 사용된다
- ✓ 여러 개의 Data Set을 동시에 만들 경우에 사용할 수 있다.

## EXAMPLE

```
DATA male female;
```

```
SET survey;
```

```
IF sex = 'Male' THEN OUTPUT male;
```

```
ELSE OUTPUT female;
```

```
RUN;
```

# 단순 반복

OUTPUT을 넣어주면 최종값이 아닌 각 단계의 결과를 모두 출력

예) 10년 동안 2000원을 투자하여 연이율 7.5%일 때, 이자수입 계산

```
DATA b;  
value=2000;  
DO i=1 to 10;  
    interest=value*0.075;  
    value +interest;  
OUTPUT;  
END; RUN;  
PROC PRINT; RUN;
```

OBS	value	i	interest
1	4122.06	11	287.586

OBS	value	i	interest
1	2150.00	1	150.000
2	2311.25	2	161.250
3	2484.59	3	173.344
4	2670.94	4	186.345
5	2871.26	5	200.320
6	3086.60	6	215.344
7	3318.10	7	231.495
8	3566.96	8	248.857
9	3834.48	9	267.522
10	4122.06	10	287.586

# 반복 처리(Do Loop)

## ■ DO문

- ✓ 지정된 반복 횟수 또는 지정한 조건이 만족될 때까지 루프 내의 문장들을 실행함
- ✓ 다음의 작업을 위해 주로 사용함
  - 반복된 작업 수행
  - 데이터의 생성
  - 데이터의 읽기
- ✓ 사용되는 형태
  - DO-END문
  - DO-WHILE-END문
  - DO-UNTIL-END문
  - DO-OVER-END문

# 반복 처리(Do Loop)\_다른 유형

## ■ DO-END문

```
DO 인덱스변수=item-1 <,...item-n>;  
반복할 SAS 문장들.....  
END;
```

### ✓ 값 리스트 지정

- do month='JAN', 'FEB', 'MAR';
- do Odd=1,3,5,7,9;
- do i=Var1, Var2, Var3;
- do j=Begindate to Today() by 7;

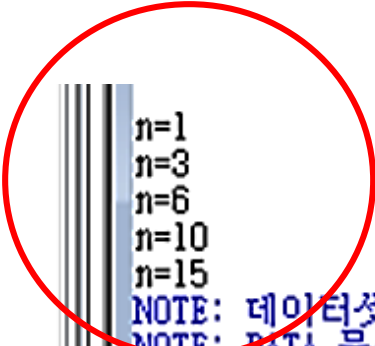
### ✓ 반복되는 값에 숫자 대신 문자(인용부호를 사용)가 들어갈 수도 있음

### ✓ 다만 따옴표 속의 문자는 네 자를 넘어서는 안 됨



# 반복 처리(Do Loop)\_예제 1

```
DATA work;  
  n=0;  
  do i=1 to 5;  
    n+i;  
  put n=;  
end;  
RUN;
```



```
n=1  
n=3  
n=6  
n=10  
n=15
```

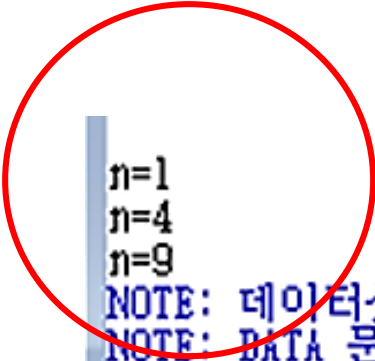
NOTE: 데이터셋 WORK.WORK은(는) 1개의 관측치와 2개의 변수를 가지고 있습니다.

NOTE: DATA 문장 실행:

실행 시간	0.10 초
cpu 시간	0.06 초

## 반복 처리(Do Loop)\_예제2

```
DATA work;  
  n=0;  
  do i=1 to 5 by 2;  
    n+i;  
    put n=;  
  end;  
RUN;
```



```
n=1  
n=4  
n=9
```

NOTE: 데이터셋 WORK.WORK은(는) 1개의 관측치와 2개의 변수를

NOTE: DATA 문장 실행:

실행 시간	0.01 초
CPU 시간	0.01 초

\*\*\*

# 반복 처리(Do Loop)\_조건부 실행

## ■ DO UNTIL

```
DO UNTIL(조건표현식);  
    반복할 SAS 문장들.....  
END;
```

- ✓ 조건식이 참이 될 때까지 DO END 사이의 SAS 명령문을 실행
- ✓ 조건식이 참이 되면 실행 중지

예) 매년 2000원을 저축하여 연이율 10%일 때, 50000원이 될 때 까지 몇 년이 걸리는지 계산

```
DATA c;
```

```
DO UNTIL(cap1 >= 50000);
```

```
    cap1+2000;
```

```
    cap1+cap1*0.1;
```

```
    year+1; END; RUN;
```

	cap1	year
1	53949,966717	13

# 반복 처리(Do Loop)\_조건부 실행

예) 매년 2000원을 저축하여 연이율 10%일 때,  
저축하는 기간을 10년으로 제한하고 싶을 때, 10년 동안의  
금액이 얼마나 될까??

```
DATA c1;
```

```
DO i=1 to 10 UNTIL(cap2>=50000);
```

```
    cap2+2000;
```

```
    cap2+cap2*0.1;
```

```
    year+1; END; RUN;
```

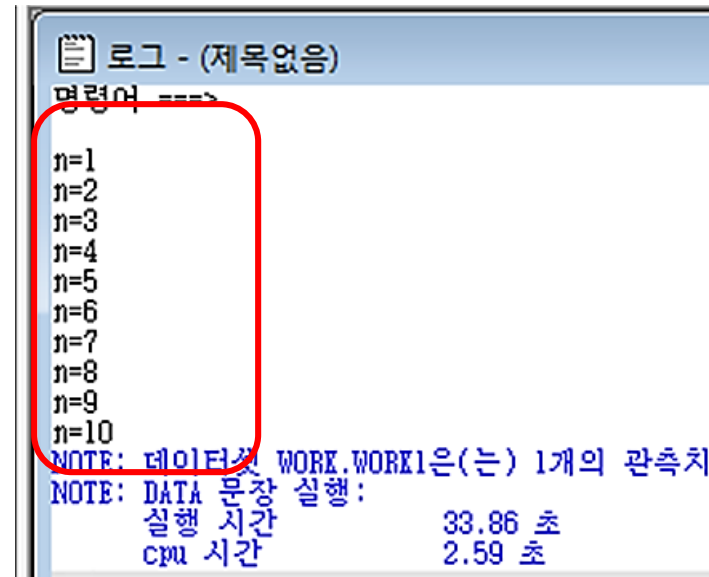
	i	cap2	year
1	11	35062,334122	10

# 반복 처리(Do Loop)\_조건부 실행

## ■ DO-UNTIL

✓ n이 10보다 크다는 조건이 성립하면 DO문을 수행하지 않음

```
Data work1;  
n=0;  
DO UNTIL (n >= 10);  
    n+ 1;  
    PUT n=;  
END ;
```



```
로그 - (제목없음)  
명령어 =====  
n=1  
n=2  
n=3  
n=4  
n=5  
n=6  
n=7  
n=8  
n=9  
n=10  
NOTE: 데이터 WORK.WORK1은(는) 1개의 관측치  
NOTE: DATA 문장 실행:  
실행 시간 33.86 초  
cpu 시간 2.59 초
```

# 반복 처리(Do Loop)\_조건부 실행

## ■ DO WHILE

```
DO WHILE(조건표현식);  
    반복할 SAS 문장들.....  
END;
```

- ✓ WHILE 다음의 조건식이 참인 경우만 DO-END 반복 수행
- ✓ 조건식이 거짓이 되면 실행되지 않음

예) DO UNTIL 예제에서 UNTIL을 WHILE로 바꿔 실행하면  
조건( $\text{cap1} \geq 50000$ )을 만족하지 않아 실행되지 않음

DATA d;

DO WHILE( $\text{cap1} \geq 50000$ );

cap1+2000;

cap1+cap1\*0.1;

year+1; END; RUN;

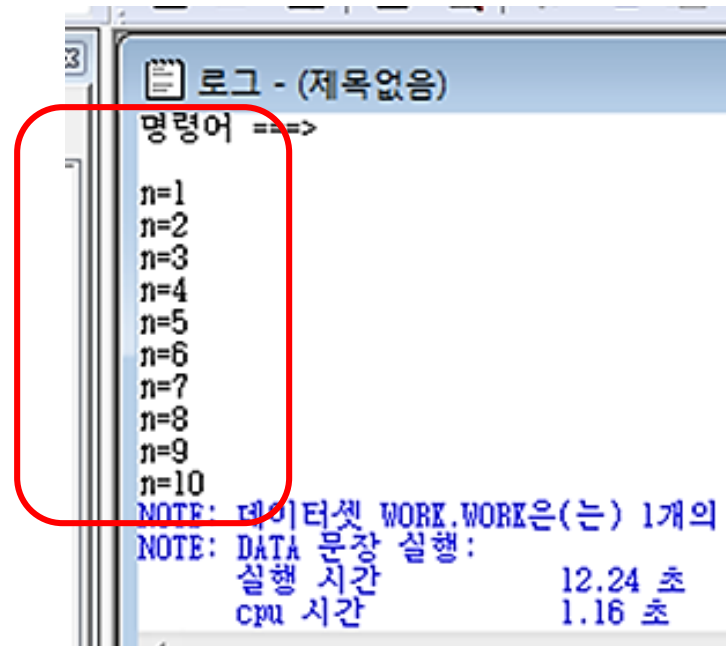
	cap1	year
1	0	0

# 반복 처리(Do Loop)\_조건부 실행

## ■ DO-WHILE

- ✓ n이 10보다 작다는 조건이 성립하는 한 DO문을 수행하는데 ,  
Log 화면에 n이 10까지 출력

```
Data Work;  
n=0;  
DO WHILE (n LT 10);  
  n+ 1;  
  PUT n=;  
END;
```



# 반복 처리(Do Loop)\_조건부 실행 예제

- ✓ 매해 1월 1일에 \$5,000씩 적금을 한다고 가정했을 때,  
목표금액 \$1,000,000을 초과하려면 몇 년이 걸리며, 그때 총 금액은 얼마인가? (이자율 복리식 년 4.5%)

```
DATA invest;
```

```
do until(Capital > 1000000);
```

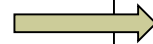
```
Year+1;
```

```
Capital+5000;
```

```
Capital+(capital*.045);
```

```
end;
```

```
RUN;
```



```
do while(Capital <= 1000000);
```

	Capital	Year
1	1029193,1704	52



## Example: DO-END 명령문(예 7.8)

아래의 데이터에서 각 행은 5개 연령대(age) 10,20,30,40,50에서  
특정 질문에 대해 조사한 사람의 수(size)와  
그 질문에 '예'라고 응답한 사람의 수(response)가 주어져 있고,  
각 연령대에 대해 여자와 남자의 정보가 구분되어 있다.  
즉 10대의 여자 40명 중 17명이 10대의 남자 30명중 11명이 '예'라고  
응답.

40	17	30	11
50	12	20	15
30	21	60	34
40	12	40	21
20	14	30	19

## Example: DO-END 명령문(예 7.8)

```
DATA survey;  
  DO age = 10 TO 50 BY 10;  
    DO sex = 'Female' , 'Male';  
      INPUT size response @@;  
      OUTPUT;  
    END;  
  END;  
  LABEL age='연령' sex='성별'  
        size='표본수' response='응답수';  
CARDS;  
40 17 30 11  
50 12 20 15  
30 21 60 34  
40 12 40 21  
20 14 30 19  
;  
RUN;  
PROC PRINT DATA=survey LABEL;  
RUN;
```

원자료는 5줄이지만  
저장된 데이터셋은  
age 5 \* sex 2 = 10개

OBS	연령	성별	표본수	응답수
1	10	Female	40	17
2	10	Male	30	11
3	20	Female	50	12
4	20	Male	20	15
5	30	Female	30	21
6	30	Male	60	34
7	40	Female	40	12
8	40	Male	40	21
9	50	Female	20	14
10	50	Male	30	19

## 5. 자동변수

- ✓작업에 필요한 정보를 임시로 저장하는 곳
- ✓일반적으로 자동 변수는 임시로 존재하며 생성된 Data Set에는 저장되지 않는다. (저장하려면 새로운 변수이름 지정하여 저장)

### • \_N\_/\_ERROR\_

- ✓SAS Data Set에 있는 각 개체/오류여부에 대하여 1부터 시작되는 자연수를 부여한다.

### • FIRST.BY와 LAST.BY

- ✓BY변수에 의하여 정렬되어 같은 값을 갖는 개체가 하나 이상 존재하는 경우 이를 개체 그룹이라 한다.
- ✓FIRST.BY는 개체 그룹의 시작을 LAST.BY는 끝을 나타내는 경우 각각 1의 값을 갖는다.
- ✓해당 변수를 PROC SORT를 이용하여 정렬한다.
- ✓DATA Step에서 BY문을 이용하여 변수를 정렬한다.
- ✓FIRST.변수 또는 LAST.변수 등의 표현을 사용한다.

# 자동변수 예 ( \_N\_ )

자동변수 \_N\_ 은 행번호를 부여 하거나 행번호를 활용한  
조건부 처리에서 유용하게 쓰임

```
data rawdata;  
  input data @@;  
cards;  
10 15 25 36  
;run;  
data rownum;  
  set rawdata;  
  idnum = _n_;  
run;
```

VIEWTABLE: Work.Rawdata

	data
1	10
2	15
3	25
4	36

VIEWTABLE: Work.Rownum

	data	idnum
1	10	1
2	15	2
3	25	3
4	36	4

## Example: 자동변수(예 7.10)

```
DATA score;
```

```
INPUT dept $ id $ score grade $;
```

```
IF _N_=1 THEN score=17; /* 15가 17로 바뀜 */
```

```
obsnum=_N_; /*자동변수를 새로운 변수로 지정한 obsnum으로 넘겨라*/
```

```
errornum=_ERROR_; /*_ERROR_ : 오류발생하면 1, 없으면 0을 변수값으로 입력*/
```

```
CARDS;
```

```
STAT S01 15 C
```

```
STAT S02 40 A
```

```
STAT S03 35 B
```

```
MATH M01 32 B
```

```
MATH M02 54 A
```

```
MATH M03 C C
```

```
MATH M04 63 A
```

```
; RUN;
```

```
PROC PRINT DATA=score; RUN;
```

OBS	dept	id	score	grade	obsnum	errornum
1	STAT	S01	17	C	1	0
2	STAT	S02	40	A	2	0
3	STAT	S03	35	B	3	0
4	MATH	M01	32	B	4	0
5	MATH	M02	54	A	5	0
6	MATH	M03	.	C	6	1
7	MATH	M04	63	A	7	0

## Example: 자동변수(예 7.11)

/\* FIRST.BY 변수 & LAST.BY 변수 : 특정 변수 별로 최대 최저 비교 \*/

**PROC SORT DATA**=score;

BY dept score; /\* dept 먼저 정렬하고 그 안에서 score 정렬 \*/

**RUN;**

**DATA** rscore;

SET score;

BY dept;

**IF FIRST.dept=1 OR LAST.dept=1;** /\* by변수의 첫번째(최저 score)와  
마지막(최고 score)을 불러와라(score 기준으로 오름차순으로 정리되었기  
때문) \*/

**RUN;**  
**PROC PRINT;**  
**RUN;**

OBS	dept	id	score	grade	obsnum	errornum
1	MATH	M03	.	C	6	1
2	MATH	M04	63	A	7	0
3	STAT	S01	17	C	1	0
4	STAT	S02	40	A	2	0

## Example: 자동변수(예 7.11)

자동변수를 데이터셋에 포함시키기 위해서는 변수이름을 다시 지정해야 한다

```
PROC SORT DATA=score;
```

```
  BY dept score; /* dept 먼저 정렬하고 그 안에서 score 정렬 */
```

```
RUN;
```

```
DATA rscore;
```

```
  SET score; f_dept=FIRST.dept; l_dept=LAST.dept;
```

```
  BY dept;
```

```
IF FIRST.dept=1 OR LAST.dept=1;RUN;
```

```
PROC PRINT;
```

```
RUN;
```

OBS	dept	id	score	grade	obsnum	errornum	f_dept	l_dept
1	MATH	M03	.	C	6	1	1	0
2	MATH	M04	63	A	7	0	0	1
3	STAT	S01	17	C	1	0	1	0
4	STAT	S02	40	A	2	0	0	1

## 6. 외부파일로 데이터 출력하기

- FILE과 PUT 명령문을 이용한 자료 출력 (예7.12)

```
DATA grade;
```

```
    INPUT name $ dept $ math stat eng kor art;
```

```
CARDS;
```

```
김철수 Stat 5 5 1 2 1
```

```
최민지 Stat . 3 1 4 5
```

```
이영희 Stat 1 5 3 2 .
```

```
오인수 Stat 4 1 2 4 .
```

```
강민호 Econ 3 2 3 1 4
```

```
;
```

```
RUN;
```

```
DATA _NULL_;
```

```
    SET grade;
```

```
    FILE 'C:\work\mysasdat\scoreout.txt'; /* FILE LOG */
```

```
    IF dept='Stat';
```

```
    PUT name 6. ' ' @11 math ' ' stat ' ' +3 eng ' ' kor ' ' art;
```

```
RUN;
```

**FILE LOG;**

```
김철수, 5,5, 1,2,1
최민지, .,3, 1,4,5
이영희, 1,5, 3,2,
오인수, 4,1, 2,4,
```

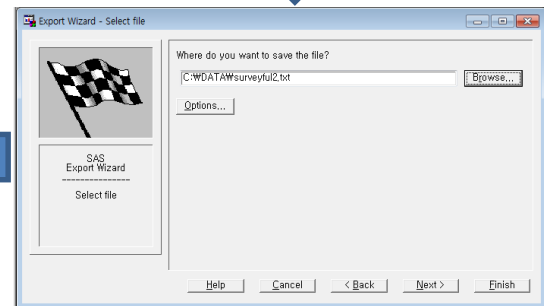
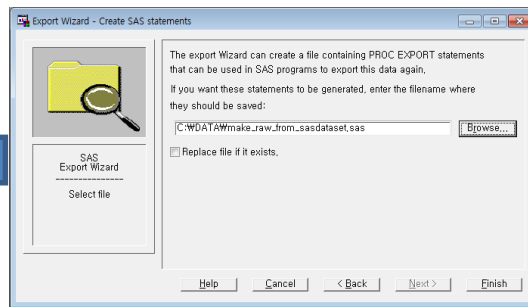
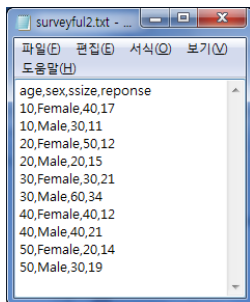
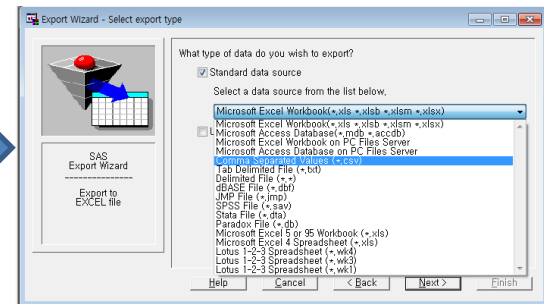
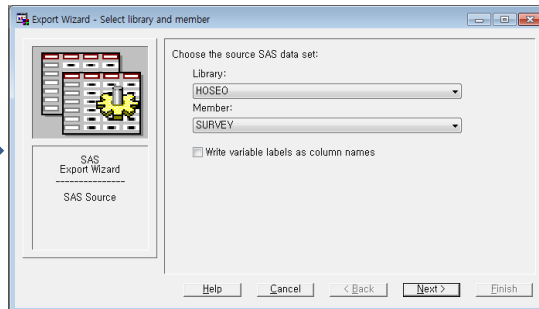
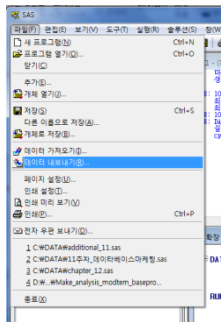


# 7. 데이터 내보내기 마법사

파일 > 데이터내보내기

데이터지정: hoseo.survey

파일형태지정 : .csv



저장파일확인

프로그램소스저장 : make\_raw.sas

파일경로/이름 지정 : surveyfull2.txt