추가 노트 (주제별)

p.2-p.19 & p.33-p.41

(pp.13-15, pp.36-41, pp.2-12, pp.16-19, pp.33-35)

1. 데이터 입력(input)과 출력(output) 제어

- OUTPUT 문장의 개념
- 한 개의 DATA 단계에서 여러 개의 데이터 생성 방법
- SELECT 구룹에 대한 이해
- 컬럼 선택 및 부분적으로 읽어올 관측치 선택 방법

2. 데이터 요약 (summarizing data)

- 변수값에 대한 총 누적값 구하는 방법
- 데이터의 구룹에 대해 총 누적값 구하는 방법

3. 원시자료(raw data) 파일 읽기

- 일정한 format 형태로 구성된 원시자료 읽기
- 원시 자료 읽을 때 record control 방법
- 구분자로 분류된 원시자료 읽기

4. 데이터 변환 (data transformation)

- SAS 함수 및 사용법

5. 데이터 반복 처리

- 데이터의 반복 처리 작업
- 같은 속성의 변수들에 대한 동일한 작업

6. 데이터 구조 변경

- 데이터 전치(transpose) 방법

7. 데이터 결합 (combining data)

- 데이터의 결합

8. Macro 언어

1. 데이터 입력(input)과 출력(output) 제어

OUTPUT 문장

: 암묵적으로 DATA 단계 가장 하단에서 출력 데이터에 생성되는 관측치에 대해, DATA 단계의 원하는 시점에서 명시적으로 관측치 생성 (데이터셋에 생성되는 관측치 제어)

(예) 어느 회사의 6개 부서의 성장률에 관한 자료이다. 향후 2년 동안 각 부서(department)가 추측된 성장률(increase)로 커진다면 매년 말 각 부서의 인원수(total_employees)는 얼마인가?

```
DATA growth;
   INPUT department $20. total_employees increase;
   FORMAT increase 5.2;
   DATALINES;
Administration          34   0.25
Engineering              9   0.30
IS                      25   0.10
Marketing               20   0.20
Sales                  201   0.30
Sales Management        11   0.10
RUN;
DATA forecast;
   SET growth;
   year=1;
   total_employees=total_employees*(1+increase);
   OUTPUT;  /* OUTPUT forecast */
   year=2;
   total_employees=total_employees*(1+increase);
   OUTPUT;  /* OUTPUT forecast */
RUN;
```

⇒

VIEWTABLE: Work.Forecast

	department	total_employees	increase	year
1	Administration	42.5	0.25	1
2	Administration	53.125	0.25	2
3	Engineering	11.7	0.30	1
4	Engineering	15.21	0.30	2
5	IS	27.5	0.10	1
6	IS	30.25	0.10	2
7	Marketing	24	0.20	1
8	Marketing	28.8	0.20	2
9	Sales	261.3	0.30	1
10	Sales	339.69	0.30	2
11	Sales Management	12.1	0.10	1
12	Sales Management	13.31	0.10	2

DATA *SASdata1 … SASdatan* ;

OUTPUT *SASdata1 … SASdatan* ;

: Creating multiple SAS data sets

(예) 한 DATA 단계에서 조건에 따라 usa, australia, other 라는 3개의 데이터셋을 한꺼번에 생성?

```
DATA usa australia other;
    SET employee; /* country, age, gender, edu 4개 변수 포함 */
    IF country='AU' THEN OUTPUT australia;
        ELSE IF country='US' THEN OUTPUT usa;
        ELSE OUTPUT other;
RUN;
```

VIEWTABLE: Dataset.Employee

	country	age	gender	edu
1	US	35	F	1
2	uS	40	F	1
3	us	43	M	1
4	AU	42	M	2
5	au	39	M	1
6	AU	45	F	1
7	KO	38	F	1
8	Ko	46	M	1
9	JA	48	F	1

⇒

VIEWTABLE: Work.Australia

	country	age	gender	edu
1	AU	42	M	2
2	AU	45	F	1

VIEWTABLE: Work.Usa

	country	age	gender	edu
1	US	35	F	1

VIEWTABLE: Work.Other

	country	age	gender	edu
1	uS	40	F	1
2	us	43	M	1
3	au	39	M	1
4	KO	38	F	1
5	Ko	46	M	1
6	JA	48	F	1

[주의] 문자값 지정 시 대소문자, 빈칸 등 정확히 구분 필요

SELECT 그룹 (끝에 END 문장 필요)

: Creating multiple SAS data sets

IF~ THEN~ ELSE 와 같은 조건 처리문

(예) 한 DATA 단계에서 조건에 따라 usa, australia, other 라는 3개의 데이터셋을 한꺼번에 생성?

```
DATA usa australia other;
   SET employee;
   SELECT (country);
         WHEN ('US') OUTPUT usa;
         WHEN ('AU') OUTPUT australia;
         OTHERWISE OUTPUT other;
   END;
RUN;
```

⇔

```
DATA usa australia other;
   SET employee;
   SELECT;
         WHEN (country='US') OUTPUT usa;
         WHEN (country='AU') OUTPUT australia;
         OTHERWISE OUTPUT other;
   END;
RUN;
```

[참고] SELECT 그룹을 사용할 때에는 빈도가 큰 것부터 빈도가 작은 것 순으로 조건을 주는 것이 **효율적**이다.

(예) 위의 예제에서,

```
SELECT (country);
      WHEN ('US', 'us', 'Us', 'uS') OUTPUT usa;
      WHEN ('AU', 'au', 'Au', 'aU') OUTPUT australia;
      OTHERWISE OUTPUT other;
END;
```

⇔

```
SELECT (UPCASE(country));
      WHEN ('US') OUTPUT usa;
      WHEN ('AU') OUTPUT australia;
      OTHERWISE OUTPUT other;
   END;
```

⇒

VIEWTABLE: Work.Usa

	country	age	gender	edu
1	US	35	F	1
2	uS	40	F	1
3	us	43	M	1

VIEWTABLE: Work.Australia

	country	age	gender	edu
1	AU	42	M	2
2	au	39	M	1
3	AU	45	F	1

VIEWTABLE: Work.Other

	country	age	gender	edu
1	KO	38	F	1
2	Ko	46	M	1
3	JA	48	F	1

(예) SELECT 구룹에서 DO-END 를 이용하여 여러 개의 문장 실행 가능

```
DATA usa australia other;
        SET employee;
        SELECT (UPCASE(country));
                WHEN ('US') DO;
                                Benefits=1;
                                OUTPUT usa;
                               END;
                WHEN ('AU') DO;
                                Benefits=2;
                                OUTPUT australia;
                               END;
                OTHERWISE DO;
                                Benefits=0;
                                OUTPUT other;
                               END;
        END;
RUN;
```

SASdataset (DROP= *var1 var2 ...*);

SASdataset (KEEP= *var1 var2 ...*);

: DATA 문장에서 또는 SET 문장에서의 **옵션**으로, 변수 선택 옵션
INPUT data set, OUTPUT data set에 모두 사용 가능

DROP *var1 var2 ...* ;

KEEP *var1 var2 ...* ;

: DATA 단계(step)에서 사용할 수 있는 **문장**으로, 변수 선택 기능
(PROC 단계에서는 사용 불가)

(예) DATA usa(DROP=country edu) australia(DROP=country) other ;
/* ⇔ DATA usa(KEEP=age gender) australia(KEEP=age gender edu) other */
SET employee;
IF country='US' THEN OUTPUT usa;
ELSE IF country='AU' THEN OUTPUT australia;
ELSE OUTPUT other;
RUN;

(예) DATA australia_new;
SET australia;
DROP country; /* SET 문, DROP 문 순서 바꿔도 동일 */
RUN;
⇒

OBS	age	gender	edu
1	42	M	2
2	39	M	1
3	45	F	1

(예) DATA australia_new;
SET australia(DROP=country); /* (cf) DATA 문장 등에서 사용 */
RUN; /* 이 예제의 경우에 결과는 위와 동일한 australia_new 생성 */
⇒

OBS	age	gender	edu
1	42	M	2
2	39	M	1
3	45	F	1

[참고] 현 DATA 단계 내에서 country 라는 변수를 이용한 processing (계산 등)이 필요 없다면 메모리 공간 차원에서 효율적
(∵ **australia 라는 데이터셋을 불러올 때 아예 country 라는 변수는 drop 하고** 가져오므로)

(예) 다음 두 결과 (australia_1, australia_2) 비교

```
DATA australia_1;  /* DROP 문장 사용 */
      SET australia;
      DROP country;
      IF country='au' THEN DELETE;
RUN;
```

⇒

OBS	age	gender	edu
1	42	M	2
2	45	F	1

```
DATA australia_2;  /* SET 문장에서 DROP 옵션 사용 */
      SET australia (DROP=country);
      IF country='au' THEN DELETE;
RUN;
```

⇒

OBS	age	gender	edu	country
1	42	M	2	
2	39	M	1	
3	45	F	1	

[참고] australia 라는 데이터셋을 불러올 때 이미 country 라는 변수는 drop 되었기 때문에 country 라는 변수를 이용한 processing 불가 (country 라는 변수는 없던 변수이므로 모두 결측치)

[참고] DROP=/KEEP= 옵션은 input/output data set에 모두 사용 가능
(e.g.) PROC PRINT DATA=australia(DROP=edu); RUN;

[참고] DROP 문장/KEEP 문장은 PROC 단계에서 사용 불가

SASdataset (FIRSTOBS= *n*);

: 읽을 데이터 시작 관측치(*n*) 지정

SASdataset (OBS= *n*); 또는 *SASdataset* (OBS=MAX);

: 읽을 데이터 끝 관측치 지정 (몇 개 관측치 사용할지 지정하는 것 아님)

아무 옵션 없으면, FIRSTOBS=1, OBS=MAX

[참고] DATA 단계, PROC 단계에서 모두 사용 가능

(예) DATA new;

```
SET employee (FIRSTOBS=2 OBS=5);
IF UPCASE(country)='AU';
/* (cf) WHERE UPCASE(country)='AU';
        WHERE와 함께 사용하면 WHERE 기준 먼저 적용한 후
  → FIRSTOBS → OBS 순으로 적용 */ /* (cf) IF 는 다름 */
RUN;   PROC PRINT DATA=new; RUN;
```

⇒

OBS	country	age	gender	edu
1	AU	42	M	2
2	au	39	M	1

(예) PROC PRINT DATA=employee(FIRSTOBS=2 OBS=5);

```
RUN;
```

⇒

OBS	country	age	gender	edu
2	uS	40	F	1
3	us	43	M	1
4	AU	42	M	2
5	au	39	M	1

original 관측치 번호

(예) DATA one;

```
INFILE 'C:₩emps.txt' FIRSTOBS=11 OBS=100;
INPUT empid 8. empname $40. country $;
RUN;
```

(예) PROC PRINT DATA=employee (OBS=2);

```
WHERE UPCASE(country)='KO'; /* WHERE와 함께 사용하면
WHERE 기준 먼저 적용한 후 → FIRSTOBS → OBS 순으로 적용 */
* WHERE UPCASE(country)="AU";
VAR age gender country;
RUN;
```

⇒

OBS	age	gender	country
7	38	F	KO
8	46	M	Ko

2. 데이터 요약 (summarizing data)

RETAIN *variable_name <initial_value>* ... ;

: 할당문을 통해 생성된 변수는 데이터 단계의 각 반복작업 시 결측치로 초기화되는데, RETAIN 문장을 사용하면 데이터 단계의 맨 처음 부분을 만나더라도 결측치로 초기화되는 것을 막아주는 기능

(누적 total 변수 생성 등 관련)

(예)
```
DATA one;
   INPUT SaleAmt @@;
   DATALINES;
     534 827 298 387 276 90
RUN;
DATA two;
   SET one;
   RETAIN CumSale 0 ;    *<<=== ;
   CumSale=CumSale+SaleAmt;
     /* CumSale 초기값 지정해주지 않으면 결측치로 처리 */
RUN;
PROC PRINT DATA=two;  RUN;
```

⇒

OBS	Sale Amt	Cum Sale
1	534	534
2	827	1361
3	298	1659
4	387	2046
5	276	2322
6	90	2412

RETAIN CumSale 0; 문장이 없다면? **(또는)** RETAIN CumSale; 이면?

⇒

OBS	Sale Amt	Cum Sale
1	534	.
2	827	.
3	298	.
4	387	.
5	276	.
6	90	.

RETAIN CumSale 0; 문장 대신 CumSale=0; 문장을 사용하면?

⇒

OBS	Sale Amt	Cum Sale
1	534	534
2	827	827
3	298	298
4	387	387
5	276	276
6	90	90

```
DATA two;
   SET one;
   RETAIN CumSale 0;
   CumSale=CumSale + SaleAmt;
RUN;
⇔  DATA two;
     SET one;
     CumSale + SaleAmt;  /* 연산자(+)를 사용했더라도 할당문에서의
       사용 결과와 달리, 합산문에서는 결측치는 무시하고 진행됨.
       즉, SaleAmt의 결측치가 없다면 위의 두 문장과 본 문장은 같은
       결과 생성*/
       /* 합산문 형태: 누적변수 + 표현식 */
    RUN;
```

[참고] 연사자 기호를 이용한 표현식 중에서 하나라도 결측치가 있으면 return되는 값도 결측치로 return 된다.
즉, CumSale=CumSale+SaleAmt; 문장처럼 연산자 기호(+)를 이용하는 경우, 결측치가 하나라도 있으면 결과값도 결측치로 나오는데, **SUM 함수**를 이용하면 **결측치를 ~~0으로 간주~~하여 무시하고** sum 결과를 출력해 준다.

SUM 함수를 사용한다면?

```
   RETAIN CumSale   ;
   CumSale=SUM(CumSale, SaleAmt);
```

- **SUM 함수 사용시** RETAIN CumSale 0; 처럼 **0 으로 초기화할 필요 없음.**
- SUM 함수 사용시 **결측치** 무시하고 sum 결과 출력.

BY *by_variable* ;

: **DATA 단계**, **PROC 단계**에서 사용 가능한 문장

(그룹별 누적 total 변수 생성 등 관련)

(예) PROC SORT DATA=*input_SAS_data_set* <OUT=*output_SAS_data_set*>;

/* input_SAS_data_set을 정렬시켜 output_SAS_data_set 이라는 새로운 데이터셋으로 저장 (OUT= 옵션을 생략하면 정렬된 데이터셋이 원래의 input_SAS_data_set 으로 재저장됨) */

BY <DESCENDING> *by_variable* ... ;

/* 디폴트 정렬방식은 오름차순인데 DESCENDING 옵션을 추가하면 내림차순으로 정렬 */

RUN;

⟾ BY 문장에 지정된 변수값에 따라 데이터 정렬

(예) DATA sale;

INFIIE 'c:₩sales.txt';

INPUT dept $ salary;

RUN;

PROC SORT DATA=sale **OUT=**salesort;

BY dept;

RUN;

PROC PRINT DATA=salesort;

RUN;

⇒

OBS	dept	salary
1	AIMIN	20000
2	AIMIN	100000
3	AIMIN	50000
4	ENGINR	25000
5	ENGINR	20000
6	ENGINR	23000
7	ENGINR	27000
8	FINANC	10000
9	FINANC	12000

DATA deptsales;

SET salesort;

BY dept; /* 지정된 변수값 별로 데이터 그룹화 가능

(단, BY 변수에 대해 미리 정렬되어 있어야 함) */

IF **FIRST.**dept=1 THEN deptsal=0; /* ⇔ **IF FIRST.dept** THEN ...;*/

/* **FIRST.dept=1** 이면(그룹이 시작될 때), deptsal 초기값은 0으로 */

deptsal+salary; /* salary 값을 누적시켜 deptsal 에 저장 */

RUN;

⇒

OBS	dept	salary	deptsal
1	ADMIN	20000	20000
2	ADMIN	100000	120000
3	ADMIN	50000	170000
4	ENGINR	25000	25000
5	ENGINR	20000	45000
6	ENGINR	23000	68000
7	ENGINR	27000	95000
8	FINANC	10000	10000
9	FINANC	12000	22000

```
DATA deptsales;
   SET salesort;
   BY dept;
   IF FIRST.dept=1 THEN deptsal=0;
   deptsal+salary;
   IF LAST.dept; /* ⇔ IF LAST.dept=1 ; */
      /* 사실이 아니면 이후 작업 계속하지 않고 (즉, output 생성되지
         않고) DATA 단계 처음으로 돌아감 */
      /* 이 문장을 추가하면 dept 그룹에서 마지막 자료만 output으로
         생성 (즉, 각 dept 별 누적 salary 만 출력 */
RUN;
```

⇒

OBS	dept	salary	deptsal
1	ADMIN	50000	170000
2	ENGINR	27000	95000
3	FINANC	12000	22000

[참고] DATA 단계에서 BY 문장은 다음 두 개의 **임시 변수들**을 생성한다.
(최종 데이터셋에는 생성 안됨)

FIRST.*by_variable*
LAST.*by_variable*

FIRST.*variable* = 1 if BY 그룹에서 첫 번째 관측치
= 0 o/w
LAST.*variable* = 1 if BY 그룹에서 마지막 관측치
= 0 o/w

(e.g.) BY dept ;
⟹ FIRST.dept
LAST.dept

3. 원시자료(raw data) 파일 읽기

INFILE *raw_data_file_name* ;

: INPUT 문장과 함께 원시자료(raw data) 읽을 수 있는 문장.

INPUT *specifications* ;

: 원시자료(raw data) 파일에 있는 값들이 어떻게 배열되어 있는지 지정하여 SAS 변수 값을 할당.
(할당 방식에는 3가지 방식이 있음)

▸ **List Input 방식 (자유/목록입력 방식)**

: **빈칸**이나 **다른 구분자(delimiter)**로 구별된 표준 데이터와 비표준 데이터 형식의 경우 사용.
구분자에 대한 정보는 **INFILE 문장에서 DLM= 옵션**으로 지정.
(특정 구분자로 되어 있는 경우, 여러 구분자 지정 가능)

(예) INFILE 'c:₩test.txt' DLM=',';/* , 로 구분 (또는 DELIMITER=) */
INPUT cust_type offer_date $ item $ discount $;

(예) INFILE 'c:₩one.txt' EXPANDTABS;
/* tab 문자 모두 빈칸으로 간주되어 tab 으로 구분된 자료 읽을 수 있음 */

▸ **Column Input 방식 (열입력 방식)**

: Fixed column에서 표준 데이터 형식의 경우 사용.
(데이터가 일정한 너비로 되어 있는 경우)
(예) INPUT type 1-4 offer $ 5-12 item $ 14-21;

▸ **Formatted input 방식 (포맷입력 방식)**

: Fixed column에서 표준 데이터와 비표준 데이터(특수문자 포함된 데이터) 형식의 경우 사용 .
데이터셋에서 모든 변수 읽을 필요없이 필요한 변수만 읽기 가능.
(데이터가 일정한 너비로 되어 있는 경우)

INPUT *pointer_control variable informat ...* ;

(예) INPUT @5 firstname $10. @20 exam 5. ;

[참고] - **표준 데이터**(standard data)는, 특별한 설명 없이 SAS 가 읽을 수 있는 데이터
(e.g) 58, -23, 67.23, 5.67E5, 1.2E-2 ...
- **비표준 데이터**(nonstandard data)는, 특별한 설명 없이는 SAS 가 (숫자로) 읽을 수 없는 데이터

(e.g) 5,823, 15%, $67.25, ₩500, 01/12/1999, 12MAY2006...
(이러한 숫자들은 **문자로는 읽어도** 특별한 지시 없이
숫자로는 못읽음)

<Informat 예제들> (5장 노트 p.24~ 예제에 추가)

<$>*informat*<w>.<d>

$ 는 문자변수 informat

informat 은 SAS informat 형식

w 는 input 데이터를 읽을 총 컬럼 수

d 는 숫자변수 중 소수점 자리 수

Informat	Raw Data Value	SAS Data Value
5.	12345	12345
COMMA7. DOLLAR7.	$12,345	12345
YEN8.	₩123,456	123456
COMMAX7. DOLLARX7.	$12.345 (또는 $12,345)	12345 (또는 12.345)
EUROX7.	€12.345	12345
EURO7.	€12,345	12345
PERCENT3.	15%	.15
MMDDYY6.	010160	0
MMDDYY8.	01/01/60	0
MMDDYY10.	01/01/1960	0
DDMMYY6.	311260	365
DDMMYY8.	31/12/60	365
DDMMYY10.	31/12/1960	365
DATE7.	31DEC59	-1
DATE9.	31DEC1959	-1
YYMMDD10.	1960-01-01	0

(예) 원시 자료(raw data)가 다음과 같은 경우,

1040 12/02/07 Outdoors 15%
2020 10/07/07 Golf 7%
1030 09/22/07 Shoes 10%
1030 09/22/07 Clothes 10%

⟹ INPUT @1 cust_type 4. @6 offer_date MMDDYY8.
@15 item $8. @24 discount PERCENT3.;

⇒

OBS	cust_ type	offer_ date	item	discount
1	1040	17502	Outdoors	0.15
2	2020	17446	Golf	0.07
3	1030	17431	Shoes	0.10
4	1030	17431	Clothes	0.10

[참고] offer_date 은 SAS 시스템 내에서의 특정 값으로 저장이 됨.
(1960년 1월 1일이면 0)
출력형식을 변경하려면 **FORMAT 문장** 사용.

(예) DATA 단계에서 FORMAT 문장 사용하면 출력형식이 영구적으로 fix 됨. (그러나 **변수유형은 그대로 숫자형태로** 저장됨)

```
DATA one;
   INPUT @1 cust_type 4. @6 offer_date MMDDYY8.
         @15 item $8. @24 discount PERCENT3.;
   FORMAT offer_date YYMMDD10.;
   DATALINES;
   1040 12/02/07 Outdoors 15%
   2020 10/07/07 Golf      7%
   1030 09/22/07 Shoes    10%
   1030 09/22/07 Clothes  10%
RUN;
```

⇒

OBS	cust_ type	offer_date	item	discount
1	1040	2007-12-02	Outdoors	0.15
2	2020	2007-10-07	Golf	0.07
3	1030	2007-09-22	Shoes	0.10
4	1030	2007-09-22	Clothes	0.10

(예) PROC 단계에서 FORMAT 문장 사용하면 프로시져 내에서만 임시적으로 영향.

```
PROC PRINT DATA=one;
   FORMAT offer_date date9. ;
   FORMAT discount PERCENT5. ;
     * 음수가 없더라도 음수에 대한 () 자리수도 포함시켜야 ;
  * FORMAT offer_data date9. discount percent5.;
RUN;
```

⇒

OBS	cust_ type	offer_ date	item	discount
1	1040	02DEC2007	Outdoors	15%
2	2020	07OCT2007	Golf	7%
3	1030	22SEP2007	Shoes	10%
4	1030	22SEP2007	Clothes	10%

[참고] [노트06] p.2

(예) Multiple INPUT 문장 (여러 줄에 거쳐 한 관측치 자료 입력)

```
1---5----0----5----0----5  (열 번호)
```

```
Ms.  Sue Farr
15 Harvey  Rd.
Macon, GA  31298
869-7008
Dr.  Kay B. Cox
163 McNeil Pl.
Kern,  CA  93280
483-3321
```

```
DATA contacts;
      INFILE 'c:₩address.txt';
      INPUT fullname $30.; /* INPUT @1 fullname $30.; */
       /* @1 생략 가능(∵새로운 줄은 항상 첫 째 열부터 읽기 시작) */
      INPUT; /* 변수명 할당은 안되더라도 둘 째 줄 불러옴 */
      INPUT address2 $25.; /* 셋째 줄 읽어서 변수값 할당 */
      INPUT phone $8.; /* 넷째 줄 읽어서 변수값 할당 */
RUN;
```

⇒

OBS	fullname	address2	phone
1	Ms. Sue Farr	Macon, GA 31298	869-7008
2	Dr. Kay B. Cox	Kern, CA 93280	483-3321

줄 포인터 (line pointer): ***#n*** (5장 참고) 또는 **/**

(예) 줄 포인터를 이용하여 하나의 INPUT 문장으로도 가능

```
DATA contacts;
      INFILE 'c:₩address.txt';
      INPUT fullname $30. / / address2 $25. / phone $8.;
            /* / 는 줄포인터를 다음 줄로 이동 */
            /* 첫 번째 / 는 두 번째 record, 두 번째 / 는 세 번째
               record, 세 번째 / 는 네 번째 record 불러옴 */
RUN;
```

⇔ 위와 동일한 결과 생성

또는

```
      INPUT fullname $30. #3 address2 $25. #4 phone $8.;
```

줄 포인터 이동 정지: @

다른 INPUT 문장이 나올 때까지 기다림.

(예) Format 형식이 (location 값에 따라) 다른 자료 읽기

```
1---5----0----5----0----5----0   (열번호)
-------------------------------------------------------
102  USA 1-20-2007 3295.50
3034 EUR 30JAN2007 1876,30
101  USA 1-30-2007 2938.00
128  USA 2-5-2007  2908.74
1345 EUR 6FEB2007  3145,60
-------------------------------------------------------
```

```
DATA sales;
    INFILE 'c:₩sales.txt';
    INPUT saleid $4. @6 location $3. @;
    IF location='USA' THEN INPUT @10 saledate MMDDYY10.
                                 @20 amount 7.;
      ELSE IF location='EUR' THEN INPUT @10 saledate DATE9.
                                   @20 amount COMMAX7.;
RUN;
```

⇒

OBS	saleid	location	saledate	amount
1	102	USA	17186	3295.50
2	3034	EUR	17196	1876.30
3	101	USA	17196	2938.00
4	128	USA	17202	2908.74
5	1345	EUR	17203	3145.60

DLM= DSD MISSOVER : INFILE 문장에서의 옵션

DLM= : 고정너비가 아닌 특정 구분자로 분류된 자유입력/목록입력 방식에서의 옵션들

DSD : 구분자는 콤마(,)이고 **연속으로 오는 콤마는 결측으로 간주** (SAS에서는 이 옵션을 사용하지 않으면 (디폴트로) 연속으로 존재하는 구분자는 한 개의 구분자로 인식)

MISSOVER : 레코드 끝 부분에 해당하는 변수값이 결측일 때 사용

(예) 콤마(,) 라는 구분자로 분류된 원시자료

```
-------------------------------------------------------
120102,Tom,Zhou,M,108255,Sales Manager,AU
120103,Wilson,Dawes,M,87975,Sales Manager,AU
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU
-------------------------------------------------------
```

```
DATA sales;
    LENGTH firstname $12 lastname $18 gender $1 jobtitle $25
           country $2;    /* 없으면, 디폴트로 문자변수는 8자리만 */
    INFILE 'c:₩sales.txt' DLM=',';
    INPUT employeeid firstname $ lastname $ gender $ salary
          jobtitle $ country $;
RUN;
```

⇒

OBS	firstname	lastname	gender	jobtitle	country	employeeid	salary
1	Tom	Zhou	M	Sales Manager	AU	120102	108255
2	Wilson	Dawes	M	Sales Manager	AU	120103	87975
3	Irenie	Elvish	F	Sales Rep. II	AU	120121	26600

(예) record 끝에 결측치가 존재하는 원시자료

```
James Kvarniq,(074) 293-8126,(701) 281-8923
Sandrina Stephano, (919) 871-7830
Cornelia Krahl, (212) 891-3241, (212) 233-5413
```

```
DATA contacts;
    LENGTH name $20 phone mobile $14;
          /* LENGTH 또는 INFORMAT 문장에서 형식 지정 안해주면
             디폴트로 문자변수는 8자리만 읽음 */
    * 또는 INFORMAT name $20. phone mobile $14. ;
    INFILE 'c:₩phone.txt' DLM=',' MISSOVER;
    INPUT name $ phone $ mobile $;
RUN;
```

⇒

OBS	name	phone	mobile
1	James Kvarniq	(074) 293-8126	(701) 281-8923
2	Sandrina Stephano	(919) 871-7830	
3	Cornelia Krahl	(212) 891-3241	(212) 233-5413

(예) 결측치가 존재하는 원시자료

```
James Kvarniq, (704) 293-8126, (701) 281-8923
Sandrina Stephano,, (919) 271-4592
Karen Ballinger,, (714) 644-9090
```

```
DATA contacts;
    LENGTH name $20 phone mobile $14;
    INFILE 'c:₩phone2.txt' DSD; * Delimiter Sensitive Data ;
```

* (cf) INFILE 'c:₩phone2.txt' DLM=',';
* (cf) INFILE 'c:₩phone2.txt' DLM=',' MISSOVER;
INPUT name $ phone $ mobile $;

RUN;

⇒

OBS	name	phone	mobile
1	James Kvarniq	(704) 293-8126	(701) 281-8923
2	Sandrina Stephano		(919) 271-4592
3	Karen Ballinger		(714) 644-9090

4. 데이터 변환 (data transformation)

: 여러 가지 **함수들** 이용한 변수 생성

- manipulating character values (문자값 조정 함수)
- manipulating numeric values (숫자값 조정 함수)
- 변수 type 변환 (명시적 또는 자동 변환 방법)

Manipulating Character Values (문자값 조정 함수)

NewVar = SUBSTR (*string*, *start* <,*length*>);

string 은 문자값, 변수, 표현식 등

start 는 시작 위치

length 는 (시작한 위치부터) 추출할 길이

생략하면 시작 위치부터 나머지 모두

(예) item_code="978-1-59994-397-8";
item_type=SUBSTR(item_code, 7, 5);
⟾ item_type=59994

SUBSTR (*string*, *start* <,*length*>) = value;

(예) location="Columbus, GA 43227";
SUBSTR(location, 11, 2)="OH";
⟾ location=Columbus, OH 43227

NewVar = LENGTH (*argument*);

: LENGTH **함수**는 **문자변수**의 trailing blanks(뒷부분공백)를 제외한 문자열의 길이

(예) code1=' ABCD '; code2=' A BC D '; code3=''; code4=' ';
leng1=LENGTH(code1);
leng2=LENGTH(code2);
leng3=LENGTH(code3);
leng4=LENGTH(code4);
⟾ leng1=5, leng2=7, leng3=1, leng4=1

(예) DATA one;
INPUT code $ @@;
IF SUBSTR(code, LENGTH(code), 1)='1' THEN check='ok';
/* 변수 code 의 마지막 값 한 자리가 1 이면 check='ok' */
/* [참고] code 는 문자변수이므로 왼쪽으로 align 되어 있음 */
id=SUBSTR(code, 1, LENGTH(code)-1);
/* 변수 code 의 맨 마지막 한 자리 값만 제외한 값을 id 변수
값으로 하여 새로운 변수 id 생성 */
DATALINES;
AQI2 CCI1 CNI2 CS3 CU1 DAI2
RUN;
PROC PRINT DATA=one; RUN;
⇒

OBS	code	check	id
1	AQI2		AQI
2	CCI1	ok	CCI
3	CNI2		CNI
4	CS3		CS
5	CU1	ok	CU
6	DAI2		DAI

NewVar = UPCASE (*argument*);
: 모두 대문자로 바꾸는 함수

(예) name='hong kil-dong';
upname=UPCASE(name);
⟾ upname=HONG KIL-DONG

NewVar = LOWCASE (*argument*);
: 모두 소문자로 바꾸는 함수

(예) name='HONG Kil-DOng';
lowname=LOWCASE(name);
⟾ lowname=hong kim-dong

NewVar = PROPCASE (*argument <,delimiter(s)>*);

: **구분자를 기준으로 단어를 구분하여 단어의 첫문자를 대문자로** 바꾸고 **나머지는 소문자로** 변경하는 함수

argument 은 문자값, 변수, 표현식 등
delimiter(s) 는 단어를 분리할 구분자 기준에 대한 옵션
생략되면, 디폴트 구분자는 **빈칸 / - (. tab**

(예) name='SURF&LINK SPORTS';
pname=PROPCASE(name); /* 디폴트 구분자 모두 적용 */
pname2=PROPCASE(name, " &"); /* 구분자는 빈칸과 & */
pname3=PROPCASE('honG/gil-dong', "");
⟹ pname=Surf&link Sports
pname2=Surf&Link Sports
pname3=Hong/gil-dong

NewVar = SCAN (*string, n <,charlist>*);

: 문자변수 값 중 **구분자 사용해 특정한 단어(들) 추출**할 때 사용하는 함수

string 은 문자값, 변수, 표현식 등
n 은 *string* 으로부터 추출할 ***n*-번째 단어** 지정
음수이면, 구분자를 기준으로 맨 끝에서부터 *n*-번째 단어
string 이 *n* 단어보다 적은 경우에는 결측값
charlist 는 단어를 구분할 문자(들) 지정하는 옵션
여러 개 문자를 **구분자**로 사용 가능
첫째 문자열 이전의 구분자는 무시됨
생략되면, 디폴트 구분자는 **빈칸 . < (+ | & ! $ *) ; - / , % ^**
NewVar 은 200 bytes

(예) second=SCAN('software and services', 2, ' '); /* 공백 기준 2-번째 */
second1=SCAN('software, hardware, services', 2, ',');
second2=SCAN('software, hardware, services', 2, ', ');
⟹ second=and
second1= hardware /* 공백 포함 */
second2=hardware /* , 또는 **빈칸**도 각각 구분자로 인식 */
/* 공백 없음 (, 를 하나의 구분자로 인식) */

(예) year1=SCAN("New Year's Day, January 1st, 2007", -1);

```
                    /* 디폴트 구분자를 기준으로 맨 마지막 단어 */
year2=SCAN("New Year's Day, January 1st, 2007", 6);
                    /* 디폴트 구분자를 기준으로 6-번째 단어 */
year3=SCAN("New Year's Day, January 1st, 2007", 6, ', ');
                    /* , 또는 빈칸 모두 구분자 */
```

⟹ year1=year2=year3=2007

Position = FIND (*string, substring <,modifiers, startpos>*);

: **특정한 값**을 찾아서 그 **(시작)위치를 나타내주는** 함수
*string*에 **처음으로 나타나는** *substring* 값의 **시작 위치(숫자)**를 찾아줌
(값이 없으면 0)

string 은 문자값, 변수, 표현식 등
substring 은 *string*에서 찾을 값 지정
찾는 값이 *string* 에 없으면 0
modifiers 는 I 사용하면 대소문자 구분하지 않는다는 의미
T 사용하면 *string*과 *substring*에서 trailing blanks(뒤에 오는 빈칸들)은 무시하라는 의미
startpos 은 *string*의 어디에서부터 *substring*을 찾을지 지정

(예)
```
text='AUSTRALIA, DENMARK, US';
pos1=FIND(text, 'US');
pos2=FIND(text, ' US');
pos3=FIND(text, 'us');
pos4=FIND(text, 'us', 'I');
pos5=FIND(text, 'us', 'I', 10);
pos7=FIND("abcd abcde fg    ", "  F   ", "IT");
```

⟹ pos1=2, pos2=20, pos3=**0**, pos4=2, pos5=21, pos7=11

NewVar = TRANWRD (*source, target, replacement*);

: 기존 값의 **특정 단어를 대체 또는 제거**

source 은 변경하고 싶은 source 지정
target 은 *source*에서 변경하고 싶은 단어 지정
지정한 단어 없으면 아무 변화 없음
trailing blanks(뒤 공백)은 제거되지 않음
replacement 는 *target*을 대체할 단어 지정
trailing blanks(뒤 공백)은 제거되지 않음
NewVar 길이 미리 지정 안되었으면 디폴트로 **200**

(예) product='Luci Knit mittens, blue';
product1=TRANWRD(product, 'Luci ', 'Lucky ');
product=TRANWRD(product, 'Luci ', 'Lucky ');
⟾ product1=Lucky Knit mittens, blue
/* product1 은 디폴트로 200-byte */
product=Lucky Knit mittens, blu /* **동일 변수명으로 다시 저장** */
/* 기존 자리수보다 더 긴 결과가 나올 때에는 값이 truncation 됨 */

NewVar = COMPRESS (*source <,chars>*);
: **(빈칸 등) 특정한 값을 제거**하는 함수

source 은 문자값, 변수, 표현식 등
chars 은 *source*에서 제거할 값 지정
지정하지 않으면 *source* 에 있는 모든 **빈칸** 제거

(예) id='20 01-005 024';
new_id1=COMPRESS(id); /* ⇔ new_id1=COMPRESS(id, ' '); */
new_id2=COMPRESS(id, '-');
new_id3=COMPRESS(id, ' -'); /* 제거할 값 여러 개 지정 가능 */
firstname='Gil-Dong';
lastname='Hong ';
fullname1=lastname || ', ' || firstname;
fullname2=COMPRESS(lastname) || ', ' || firstname;
⟾ new_id1=2001-005024
new_id2=20 01005 024
new_id3=2001005024
fullname1=Hong , Gil-Dong
fullname2=Hong, Gil-Dong

NewVar = TRIM (*string*);
: 문자 변수값의 **trailing blanks(뒤 빈칸) 제거**하는 함수

(예) notrailing=TRIM('NoTrailing ') || 'Blanks';
⟾ notrailing=NoTrailingBlanks

NewVar = STRIP (*string*);
: 문자 변수값의 **trailing blanks(뒤 빈칸)와 leading blanks(앞 빈칸) 제거**하는 함수

(예) noblank=STRIP(' NoBlanks ') || ' At All';
⟾ noblank=NoBlanks At All

Manipulating Numeric Values (숫자값 조정 함수)

NewVar = ROUND (*argument* <,*round-off-unit*>);

argument 은 숫자값, 변수, 수식 등

round-off-unit 는 반올림하고자 하는 소숫점 자리 지정

생략 시 소수점 1자리에서 반올림하여 정수값 출력

(예) newvar1=ROUND(12.12);
newvar2=ROUND(42.65, .1);
newvar3=ROUND(-6.478);
newvar4=ROUND(96.47, 10);
⟾ newvar1=12, newvar2=42.7, newvar3=-6, newvar4=100

(예) num=14;
newnum=ROUND(num, 3); /* 가장 가까운 3의 배수로 반올림 */
newnum1=ROUND(12.69, 0.25); /* 가장 가까운 .25의 배수로 반올림 */
/* 12.50 과 12.75 중 더 가까운 값 */
newnum2=ROUND(42.65, 0.5); /* 가장 가까운 .5의 배수로 반올림 */
/* 43 과 42.5 중 더 가까운 값 */
⟾ newnum=15, newnum1=12.75, newnum2=42.5

NewVar = CEIL (*argument*);

argument 보다 **크거나 같은 최소 정수** 산출 함수

argument 은 숫자값, 변수, 수식 등

(예) x=CEIL(4.4);
y=CEIL(-3.6);
⟾ x=5, y=-3

NewVar = FLOOR (*argument*);

argument 보다 **작거나 같은 최대 정수** 산출 함수

argument 은 숫자값, 변수, 수식 등

(예) x=FLOOR(3.6);
y=FLOOR(-4.3);
⟾ x=3, y=-5

NewVar = INT (*argument*);

정수 산출 함수

argument 은 숫자값, 변수, 수식 등

(예) x=INT(3.6);
y=INT(-4.3);
⟾ x=3, y=-4

SUM 함수

합계 산출 함수
(**결측치는 무시**하고 계산)

(예) total=SUM(score1, score2, score3, score4);
⇔ total=SUM(OF score1-score4);
⇔ total=SUM(OF score1--score4);
/* 데이터셋에 저장되어 있는 변수명 순서대로 */
⇔ total=SUM(OF score:);
/* score로 시작하는 변수가 이 4개 변수만 있다면 */
/* SUM(score:) 는 안됨 */

(예) var1=12; var2=.; var3=7; var4=5;
sumvar=SUM(var1, var2, var3, var4);
totalvar=var1+var2+var3+var4;
⟾ sumvar = 24
totalvar = .

MEAN 함수

평균 산출 함수
(**결측치는 무시**하고 계산)

(예) var1=12; var2=.; var3=7; var4=5;
avgvar=MEAN(OF var1-var4);
meanvar=(var1+var2+var3+var4)/4;
⟾ avgvar = 8 (=(12+7+5)/3)
meanvar = .

MIN 함수

최소값 산출 함수

(결측치는 무시)

MAX 함수

최대값 산출 함수

(결측치는 무시)

N 함수

비결측 숫자 갯수 산출 함수

(예) var1=12; var2=.; var3=7; var4=5;

nvar=**N**(OF var:); /* var 로 시작하는 결측이 아닌 모든 변수의 갯수 */

⟾ nvar=3

(예) v1='a'; v2=1; v3=.; v4=''; v5=2;

nnvar=**N**(OF v1-v5);

⟾ nnvar=2

NMISS 함수

결측 숫자 개수 산출 함수

(예) var1=12; var2=.; var3=7; var4=5;

nmissvar=**NMISS**(OF var1-var4);

⟾ nmissvar=1

(예) v1='a'; v2=1; v3=.; v4=''; v5=2;

nmiss=**NMISS**(OF v1-v5);

⟾ nmiss=3

CMISS 함수

결측 숫자 또는 결측 문자 개수 산출 함수 (SAS9.2 ~)

(예) var1='a'; var2=''; var3='c'; var4='d';

cmissvar=**CMISS**(OF var1-var4);

⟾ cmissvar=1

(예) v1='a'; v2=1; v3=.; v4=' '; v5=2;

cmiss=**CMISS**(OF v1-v5);

⟾ cmiss=2

변수 Type 변환 - 명시적 변환 방법
- 자동 변환 방법

- 명시적 변환 방법:
문자변수를 --> 숫자변수로

NumVar = INPUT (*source*, *informat*);

(예)
```
cvar1='32000';              /* Type=Char, Length=5 */
cvar2='32.000';             /* Type=Char, Length=6 */
cvar3='03may2008';          /* Type=Char, Length=9 */
cvar4='030508';             /* Type=Char, Length=6 */
nvar1=INPUT(cvar1, 5.);
nvar2=INPUT(cvar2, COMMAX6.);
nvar3=INPUT(cvar3, DATE9.);
nvar4=INPUT(cvar4, DDMMYY6.);
```
⟹ nvar1=32000 (Type=Num, Length=8)
nvar2=32000 (Type=Num, Length=8)
nvar3=17655 (Type=Num, Length=8)
nvar4=17655 (Type=Num, Length=8)

(예)
```
DATA one;
   shareprice='$132.25';    /* Type=Char, Length=7 */
   myshares=125;
   totalvalue=INPUT(shareprice, COMMA7.)*myshares;
RUN;
```
⟹ totalvalue=16531.25 (<<= 132.25x125)

(예)
```
DATA hrdata;
   INPUT ID $5. GrossPay $6. @13 Hired $10. @24 Mobile $8.
         Code;
   DATALINES;
36    52,000 04/13/2004 393-0956 303
48    32,000 08/25/2006 770-8292 919
52    49,000 06/08/2005 449-5239 301
RUN;
```

```
DATA new;
   SET hrdata;
   EmpID=INPUT(ID, 5.)+11000;
   Bonus=INPUT(GrossPay, COMMA6.)*0.1;
   HireDate=INPUT(Hired, MMDDYY10.);
RUN;
```

⟹

ID	Gross Pay	Hired	Mobile	Code	EmpID	Bonus	Hire Date
36	52,000	04/13/2004	393-0956	303	11036	5200	16174
48	32,000	08/25/2006	770-8292	919	11048	3200	17038
52	49,000	06/08/2005	449-5239	301	11052	4900	16595

```
PROC PRINT DATA=new NOOBS;
   VAR EmpID GrossPay Bonus HireDate;
   FORMAT HireDate MMDDYY10.;
RUN;
```

⟹

EmpID	Gross Pay	Bonus	HireDate
11036	52,000	5200	04/13/2004
11048	32,000	3200	08/25/2006
11052	49,000	4900	06/08/2005

(예) GrossPay=INPUT(GrossPay, COMMA6.)*0.1;

⟹ GrossPay 는 여전히 **문자변수** 의 Length=6

동일한 변수명으로 변수 type을 변경하는 것은 불가능

(∵ 변수방이 이미 문자 변수로 생성되어 있으므로)

다음과 같은 방법으로는 가능

1) 입력데이터의 변수명 변경한 후
2) INPUT(또는 PUT) 함수를 통한 값 속성 변경하고
3) 변환된 최종 변수 선택

```
(e.g.) DATA new (DROP=CharGross);
          SET hrdata (RENAME=(GrossPay=CharGross) );
          EmpID=INPUT(ID, 5.)+11000;
          GrossPay=INPUT(CharGross, COMMA6.)*0.1;
             /* GrossPay 는 숫자변수 (length=8) */
          HireDate=INPUT(Hired, MMDDYY10.);
       RUN;
```

SAS-data-set **(RENAME= (***old-name = new-name***))** ;

- 명시적 변환 방법:

숫자변수를 --> 문자변수로

CharVar = PUT (*source*, *format*);

(예) DATA conversion;

nvar1=614;

nvar2=55000;

nvar3=366;

cvar1=PUT(nvar1, 3.);

cvar2=PUT(nvar2, DOLLAR7.);

cvar3=PUT(nvar3, DATE9.);

RUN;

PROC CONTENTS DATA=conversion VARNUM; RUN;

⟹

#	변수	유형	길이
1	nvar1	수치	8
2	nvar2	수치	8
3	nvar3	수치	8
4	cvar1	문자	3
5	cvar2	문자	7
6	cvar3	문자	9

PROC PRINT DATA=conversion NOOBS; RUN;

⟹

nvar1	nvar2	nvar3	cvar1	cvar2	cvar3
614	55000	366	614	$55,000	01JAN1961

(예) DATA new;

INPUT code mobile $ @@;

phone='(' || PUT(code, 3.) || ') ' || mobile;

DATALINES;

303 393-0956 919 770-8292 301 449-5239

RUN;

⟹

code	mobile	phone
303	393-0956	(303) 393-0956
919	770-8292	(919) 770-8292
301	449-5239	(301) 449-5239

- 자동 변환 방법:

문자변수를 --> 숫자변수로

(예) DATA one @@;

INPUT ID $5.;

EmpID=ID+11000;

DATALINES;

36 48 52

RUN;

⟹

ID	EmpID
36	11036
48	11048
52	11052

※ 명시적 변환방법과는 달리, 자동 변환 방법은 로그창에 문자변수가 숫자변수로 변환 되었다는 메시지 출력됨

```
NOTE: 다음의 위치에서 문자형 값이 숫자형 값으로 변환되었습니다: (행):(칼럼)
      243:10
```

(예) DATA hrdata;

INPUT ID $5. GrossPay $6. @13 Hired $10. @24 Mobile $8.

Code;

DATALINES;

36 52,000 04/13/2004 393-0956 303

48 32,000 08/25/2006 770-8292 919

52 49,000 06/08/2005 449-5239 301

RUN;

DATA new;

SET hrdata;

bonus=GrossPay * 0.1;

RUN;

⟹

ID	Gross Pay	Hired	Mobile	Code	bonus
36	52,000	04/13/2004	393-0956	303	.
48	32,000	08/25/2006	770-8292	919	.
52	49,000	06/08/2005	449-5239	301	.

※ 문자변수가 숫자변수로 자동 변환될 경우, 표준숫자 형태(소숫점, sign, E-notation)가 아니면($, 기호 등) 결측으로 return 됨.

※ 비표준 숫자값인 경우에는, 자동 변환이 아닌, 명시적 변환 필요.

(e.g.) bonus=INPUT(GrossPay, COMMA6.) * 0.1;

- 자동 변환 방법:

숫자변수를 --> 문자변수로

(예) DATA new;

```
INPUT code mobile $ @@;
phone='(' || code || ') ' || mobile;
      /* code=숫자변수    mobile=문자변수 */
/*23 =  1 +   12   +   2  +  8 */
DATALINES;
  303 393-0956 919 770-8292 301 449-5239
RUN;
```

⟹

code	mobile		phone
303	393-0956	(	303) 393-0956
919	770-8292	(	919) 770-8292
301	449-5239	(	301) 449-5239

9 byte 빈칸

숫자변수를 문자변수로 자동변환시,

code 는 12 byte 숫자변수 (BEST12. 으로 오른쪽으로 align)

(예) DATA one;

```
num1=1234;                  /* num1 은 12. */
num2=5678;                  /* num2 은 12. */
LENGTH char1 $ 10;
char1=num1;                 /* char1 은 $10. */
char2=num1 || num2;         /* char2 은 $24. */
char3=COMPRESS(num1); /* char3 은 $12. */
RUN;
```

⟹

num1	num2	char1	char2	char3
1234	5678	1234	1234 5678	1234

5. 데이터 반복처리

Do Loop Processing

```
DO index-variable=start TO stop <By increment> ;
        iterated SAS statements ...
END;
```

: 초기값, 종료값, 증분값 지정하여 사용

increment 생략 시 디폴트는 +1

```
DO index-variable=item-1 <, ... , item-n> ;
        iterated SAS statements ...
END;
```

: 일련의 항목을 열거하여 반복 횟수 지정

(예) 500만 원을 4.5%의 연이자율로 20년 동안 투자했을 때 **년 이율 (복리로) 받을 경우**와 **분기별(복리로) 받을 경우 이자액** 비교

```
DATA compound (drop=j);
   amount=5000000;
   rate=0.045;
   DO j=1 TO 20;
      yearly + ((yearly+amount) * rate);
            /* DO loop 사용 안하면 이 문장을 20번 */
      * OUTPUT; /* output 문장 추가하면 매년에 대한 yearly 값이
                   관측으로 생성 */
   END;
   DO j=1 TO 80;
      quarterly + ((quarterly + amount) * rate/4);
            /* DO loop 사용 안하면 이 문장을 80번 */
   END;
* OUTPUT; /* output 문장이 생략되어 있음 */
RUN;
```

amount	rate	yearly	quarterly
5000000	0.045	7058570.1242	7236374.8849

```
PROC PRINT DATA=compound;
      FORMAT rate PERCENT6.1 yearly quarterly YEN13.2;
RUN;
```

OBS	amount	rate	yearly	quarterly
1	5000000	4.5%	₩7,058,570.12	₩7,236,374.88

(예) DO month='Jan', 'Feb', 'Mar'; /* 문자상수 */
.....
END;

(예) DO odd=1, 3, 5, 7, 8; /* 숫자상수 */
.....
END;

(예) DO j=var1, var2, var3; /* 변수 */
.....
END;

DO WHILE (*expression*) ;
<additional SAS statements>
END;

: 조건의 참/거짓에 따라 do-loop 실행
(*expression* 조건이 참인 한 do-loop 실행)
: **한 번도 실행 안될 수 있음**
(loop 의 **처음에서** *expression* 조건 평가/판단)

DO UNTIL (*expression*) ;
<additional SAS statements>
END;

: 조건의 참/거짓에 따라 do-loop 실행
(*expression* 조건이 참일때 do-loop 실행 중지)
: **적어도 한 번은 실행됨**
(loop 의 **마지막에서** *expression* 조건 평가/판단)

(예) DATA invest;
DO UNTIL (capital > 1000000);
/* ⇔ DO WHILE (capital <= 1000000); */
year+1;
capital+5000;
capital+(capital*0.045);
END;
RUN;
⇒

capital	year
1029193.17	52

DO *index-var=start* TO *stop* <BY *increment*> WHILE | UNTIL (*expression*) ;
<additional SAS statements>
END;

: DO-loop with conditional clause

(예) DATA invest;
DO year=1 TO 30 **UNTIL**(capital>250000);
capital+5000;
capital+(capital*0.045);
END;
RUN;
PROC PRINT DATA=invest NOOBS;
FORMAT capital DOLLAR14.2;
RUN;
⇒

year	capital
27	$264,966.67

[참고] index 변수 증가 전에 condition 체크
(loop 의 **마지막에서** *expression* 조건 평가/판단)
27번 반복 작업

(예) DATA invest;
DO year=1 TO 30 **WHILE**(capital<=250000);
capital+5000;
capital+(capital*0.045);
END;
RUN;
PROC PRINT DATA=invest NOOBS;
FORMAT capital DOLLAR14.2;
RUN;
⇒

year	capital
28	$264,966.67

[참고] condition 체크 전에 index 변수 증가
(loop **처음에서** *expression* 조건 평가/판단)
27번 반복 작업

Array Processing

: 반복 계산 시, 같은 속성 갖는 많은 변수 생성 시, 데이터 읽을 때, 변수들 비교 시 사용

: **동일한 속성을 갖는 변수들에 대해 임시로 그룹핑한 개념**으로 **변수는 아님**. Array 이름으로 구분되며 array 의 각 값은 element 로 불림. **DATA 단계 실행되는 동안에만 존재. 숫자형 변수들만 또는 문자형 변수들로만 구성**되어야 하며 **혼용은 불가능**.

ARRAY *array-name*{*subscript*}<$> <*length*> <*array-elements*><(*initial-val-list*)>;

array-name 은 array 이름(SAS이름(변수명, 데이터셋명))짓는 규칙에 맞게, **단, 동일 DATA 단계 내 존재하는 변수명은 안됨)**

{*subscript*} 은 *elements* 의 수

* 지정하면 element list에 지정된 수로 **element 차원** 지정

$ 은 *elements* 가 문자라는 것 표시

length 은 *elements* 의 length

array-elements 은 *elements* 의 이름

initial-val-list 는 array 의 초기값 지정

(예) DATA charity;

```
INPUT employee_id qtr1 qtr2 qtr3 qtr4;
qtr1=qtr1*1.25;
qtr2=qtr2*1.25;
qtr3=qtr3*1.25;
qtr4=qtr4*1.25;
DATALINES;
   120265    .   .    . 25
   120267 15 15 15 15
   120269 20 20 20 20
   120270 20 10  5   .
RUN;
```

DO-loop 사용 가능?

```
DO j=1 TO 4;
   ??
END;
```

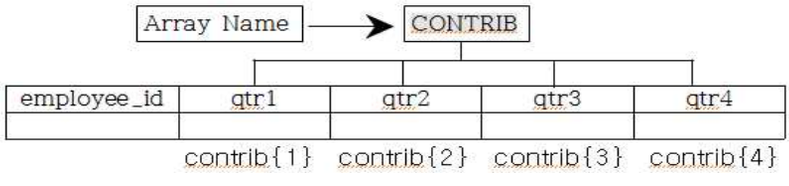
변수명 다르므로 Do-loop 내에서 하나의 작업수행으로 작업 불가능

⟾ **ARRAY 사용해야**

⇒

employee_id	qtr1	qtr2	qtr3	qtr4
120265	.	.	.	31.25
120267	18.75	18.75	18.75	18.75
120269	25.00	25.00	25.00	25.00
120270	25.00	12.50	6.25	.

ARRAY contrib{4} qtr1 qtr2 qtr3 qtr4; 문장 이용



(예) ARRAY contrib{*} qtr1 qtr2 qtr3 qtr4;
(*element-list*)

/* * 지정하면 element list에 지정된 수로 element 차원 정해짐 */

(예) ARRAY contrib{*} qtr: ;

/* qtr 로 시작하는 모든 변수를 array로 만듦.
qtr 로 시작하는 변수의 수를 모를 때 {*} 사용 */

(예) ARRAY amt{*} Q1 Q2 ThrdQ Qtr4;

/* element 이름은 유사성, 관련성 없어도 되고 sequential
아니어도 상관 없음 */

```
DO index-var = 1 TO no-of-elements-in-array ;
    <additional SAS statements>
END;
```

(예) DATA charity (DROP=j);
INPUT employee_id qtr1 qtr2 qtr3 qtr4;
ARRAY contrib{4} qtr1-qtr4;
DO j = 1 TO 4;
contrib{j} = contrib{j} * 1.25;
END;
DATALINES;
120265 . . . 25
120267 15 15 15 15
120269 20 20 20 20
120270 20 10 5 .
RUN;

⟾ j=1 일 때, contrib{1}=contrib{1}*1.25
즉, qtr1=qtr1*1.25
j=2 일 때, contrib{2}=contrib{2}*1.25
즉, qtr2=qtr2*1.25
j=3 일 때, contrib{3}=contrib{3}*1.25
즉, qtr3=qtr3*1.25
j=4 일 때, contrib{4}=contrib{4}*1.25
즉, qtr4=qtr4*1.25

⇒ 위와 동일한 결과

employee_ id	qtr1	qtr2	qtr3	qtr4
120265	.	.	.	31.25
120267	18.75	18.75	18.75	18.75
120269	25.00	25.00	25.00	25.00
120270	25.00	12.50	6.25	.

(예)

```
DATA test;
   INPUT employee_id qtr1 qtr2 qtr3 qtr4;
   ARRAY val{4} qtr1-qtr4;
   total1=SUM(OF qtr1-qtr4); /* total1 결측 아님 (∵ SUM 함수) */
   * total=qtr1+qtr2+qtr3+qtr4;  /* 결측인 경우도 있음;
   total2=SUM(OF val{*});  /* array as a function argument */
   /* ⇔ total2=SUM(OF val{1} val{2} val{3} val{4} */
   DATALINES;
      120265   .  .   . 25
      120267 15 15 15 15
      120269 20 20 20 20
      120270 20 10  5  .
RUN;
PROC PRINT DATA=test NOOBS;
   VAR employee_id total1 total2;
RUN;
```

⇒

employee_ id	total1	total2
120265	25	25
120267	60	60
120269	80	80
120270	35	35

(예) DIM 함수: **DIM(*array_name*)** : Array 차원 구해주는 함수

```
ARRAY contrib{*} qtr: ;
num_ele=DIM(contrib);
DO j=1 TO num_ele;
   contrib{j}=contrib{j}*1.25;
END;
```

(예) (숫자)변수 생성 시 ARRAY 사용

```
DATA percent (DROP=j);
   INPUT employee_id qtr1 qtr2 qtr3 qtr4;
   ARRAY contrib{4} qtr1-qtr4;
   ARRAY percent{4} percent1-percent4;
     /* percent1-percent4 의 4개 숫자변수 생성 */
     /* ⇔ ARRAY percent{4}; */
```

```
    /* element-list 생략하면 array명 뒤에 차원수만큼 순차적으로 숫자
       붙여서 변수명 생성 : percent1-percent4 의 4개 변수 생성 */
  total=SUM(OF contrib{*});     /* total 결측 아님 (∵ SUM 함수) */
  DO j=1 TO 4;
    percent{j}=contrib{j}/total; /* percent1~percent4 결측 가능 */
  END;
  DATALINES;
    120265   .  .   . 25
    120267 15 15 15 15
    120269 20 20 20 20
    120270 20 10  5  .
RUN;
```

⇒

employee_id	qtr1	qtr2	qtr3	qtr4	percent1	percent2	percent3	percent4	total
120265	.	.	.	25	.	.	.	1.00	25
120267	15	15	15	15	0.25000	0.25000	0.25000	0.25	60
120269	20	20	20	20	0.25000	0.25000	0.25000	0.25	80
120270	20	10	5	.	0.57143	0.28571	0.14286	.	35

```
PROC PRINT DATA=percent NOOBS;
  VAR employee_id percent1-percent4;
  FORMAT percent1-percent4 PERCENT6.;  /* 최소 6자리 필요 */
RUN;                                   /* PERCENTw.d format 참고 */
```

⇒

employee_id	percent1	percent2	percent3	percent4
120265	.	.	.	100%
120267	25%	25%	25%	25%
120269	25%	25%	25%	25%
120270	57%	29%	14%	.

(예) (숫자)변수 생성

```
DATA changes (DROP=j);
  INPUT employee_id qtr1 qtr2 qtr3 qtr4;
  ARRAY contrib{4} qtr1-qtr4;
  ARRAY diff{3}; /* element-list 생략되었으므로 array명 뒤에
       차원수만큼 순차적으로 숫자붙여서 변수명 생성 : diff1-diff3 */
  DO j=1 TO 3;
    diff{j}=contrib{j+1}-contrib{j};
      * j=1일 때, diff{1}=contrib{2}-contrib{1} ⇔ diff1 = qtr2-qtr1;
      * j=2일 때, diff{2}=contrib{3}-contrib{2} ⇔ diff2 = qtr3-qtr2;
      * j=3일 때, diff{3}=contrib{4}-contrib{3} ⇔ diff3 = qtr4-qtr3;
  END;
```

```
DATALINES;
   120265   .  .   . 25
   120267 15 15 15 15
   120269 20 20 20 20
   120270 20 10  5  .
RUN;
```

⇒

employee_id	qtr1	qtr2	qtr3	qtr4	diff1	diff2	diff3
120265	.	.	.	25	.	.	.
120267	15	15	15	15	0	0	0
120269	20	20	20	20	0	0	0
120270	20	10	5	.	-10	-5	.

(예) (문자)변수 생성 시 ARRAY 사용

```
ARRAY month{6} $ 10;
     /* month1-month6 의 6개 문자변수 생성 (10-문자열) */
ARRAY year{6} $;
     /* year1-year6 의 6개 문자 변수 생성 (8-문자열) */
```

(예) ARRAY 의 초기값 할당

```
DATA compare (DROP=j goal1-goal4);
   INPUT employee_id qtr1 qtr2 qtr3 qtr4;
   ARRAY contrib{4} qtr1-qtr4;
   ARRAY diff{4};
   ARRAY goal{4} (10, 20, 20, 15);  *<<<<<<<<<<<<<<<<<=== (1) ;
     /* 정해진 분기당 실적(goal) 달성했는지 check */
     /* goal 에 대한 초기값 할당 */
   DO j=1 TO 4;
     diff{j}=SUM(contrib{j}, -goal{j}); *(cf) diff{j}=contrib{j}-goal{j};
   END;
   DATALINES;
     120265   .  .   . 25
     120267 15 15 15 15
     120269 20 20 20 20
     120270 20 10  5  .
RUN;
```

⇒

employee_id	qtr1	qtr2	qtr3	qtr4	diff1	diff2	diff3	diff4
120265	.	.	.	25	-10	-20	-20	10
120267	15	15	15	15	5	-5	-5	0
120269	20	20	20	20	10	0	0	5
120270	20	10	5	.	10	-10	-15	-15

위 프로그램 (1) 에서 goal1-goal4 변수를 최종 데이터셋에는 생성하지 않으려면,

```
ARRAY goal{4} _TEMPORARY_ (10, 20, 20, 15);
  /* goal1-goal4 는 임시변수로 생성되어 별도로 drop을 하지
     않아도 최종 데이터셋에는 생성되지 않음 */
```

[참고] ARRAY 초기값 지정 시 차원수가 실제 지정한 초기값 개수보다 많으면, 나머지 array 변수값은 결측으로 처리되고,
반대의 경우에는 초과된 초기값은 무시됨

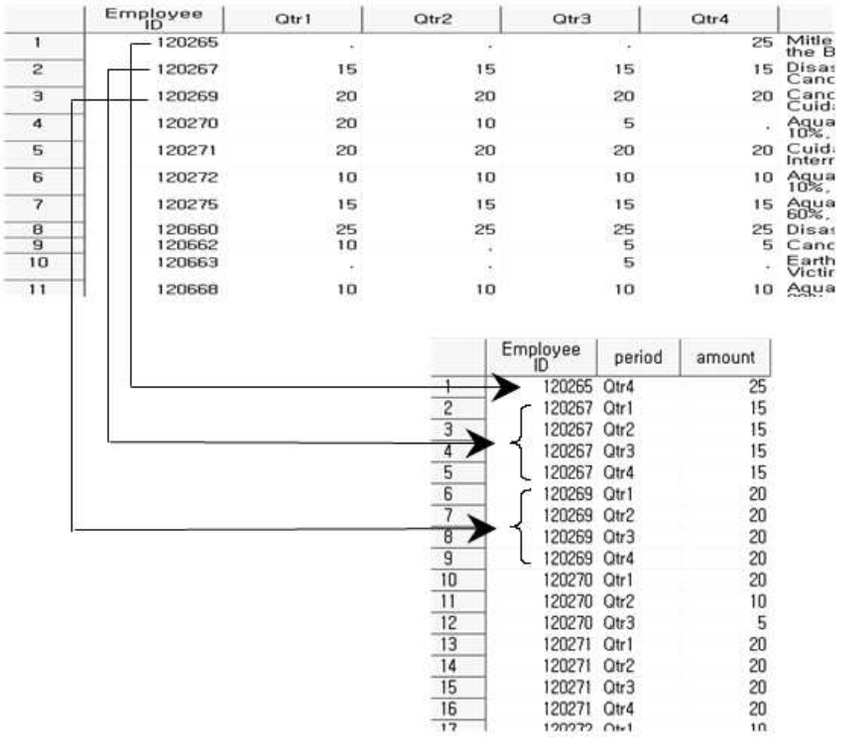
(e.g.) ARRAY test{3} (1 2);
==> test1=1, test2=2, test3=.

(e.g.) ARRAY test{3} (1 2 3 4);
==> test1=1, test2=2, test3=3

6. 데이터 구조 변경

DATA Rotating

(예) 비결측 분기 자료를 별도의 관측(observation)으로 데이터 구조 변경



	Employee ID	Qtr1	Qtr2	Qtr3	Qtr4	
1	120265	.	.	.	25	Mitle the B
2	120267	15	15	15	15	Disa: Canc
3	120269	20	20	20	20	Canc Cuid:
4	120270	20	10	5	.	Aqua 10%.
5	120271	20	20	20	20	Cuid: Interr
6	120272	10	10	10	10	Aqua 10%.
7	120275	15	15	15	15	Aqua 60%.
8	120660	25	25	25	25	Disa:
9	120662	10	.	5	5	Canc
10	120663	.	.	5	.	Earth Victir
11	120668	10	10	10	10	Aqua

	Employee ID	period	amount
1	120265	Qtr4	25
2	120267	Qtr1	15
3	120267	Qtr2	15
4	120267	Qtr3	15
5	120267	Qtr4	15
6	120269	Qtr1	20
7	120269	Qtr2	20
8	120269	Qtr3	20
9	120269	Qtr4	20
10	120270	Qtr1	20
11	120270	Qtr2	10
12	120270	Qtr3	5
13	120271	Qtr1	20
14	120271	Qtr2	20
15	120271	Qtr3	20
16	120271	Qtr4	20

```
DATA rotate;
   SET employee_donations (DROP=recipients paid_by);
            /* SAS 프로그램 연습에서는 데이터의 일부만 사용 */
   ARRAY contrib{4} qtr1-qtr4;
   DO k=1 TO 4;
     IF contrib{k} NE . THEN DO;
                              period=CATS("Qtr", k);
                              amount=contrib{k};
                              OUTPUT;
                            END;
   END;
RUN;
```

[참고] CATS 함수는 **앞, 뒤 빈칸(leading, trailing blanks) 지우고** 문자열 연결시키는 함수

CATS(*item-1 <, ..., item-n>*);

item 은 문자/숫자 상수, 변수, 표현식

item 이 숫자이면 문자로 변환됨

⇒

OBS	Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	k	period	amount
1	120265	.	.	.	25	4	Qtr4	25
2	120267	15	15	15	15	1	Qtr1	15
3	120267	15	15	15	15	2	Qtr2	15
4	120267	15	15	15	15	3	Qtr3	15
5	120267	15	15	15	15	4	Qtr4	15
6	120269	20	20	20	20	1	Qtr1	20
7	120269	20	20	20	20	2	Qtr2	20
8	120269	20	20	20	20	3	Qtr3	20
9	120269	20	20	20	20	4	Qtr4	20
10	120270	20	10	5	.	1	Qtr1	20
11	120270	20	10	5	.	2	Qtr2	10
12	120270	20	10	5	.	3	Qtr3	5
13	120271	20	20	20	20	1	Qtr1	20
14	120271	20	20	20	20	2	Qtr2	20
15	120271	20	20	20	20	3	Qtr3	20
16	120271	20	20	20	20	4	Qtr4	20
17	120272	10	10	10	10	1	Qtr1	10
18	120272	10	10	10	10	2	Qtr2	10
19	120272	10	10	10	10	3	Qtr3	10
20	120272	10	10	10	10	4	Qtr4	10
21	120275	15	15	15	15	1	Qtr1	15
22	120275	15	15	15	15	2	Qtr2	15
23	120275	15	15	15	15	3	Qtr3	15
24	120275	15	15	15	15	4	Qtr4	15
25	120660	25	25	25	25	1	Qtr1	25

```
PROC FREQ DATA=rotate;
    /* SAS 프로그램 연습에서는 데이터의 일부만 사용 */
    TABLES period / nocum nopct;
                    /* nocum: 누적값 표시 안함 */
                    /* nopct: %값 표시 안함 */
RUN;
```

⇒

FREQ 프로시저

period	빈도
Qtr1	110
Qtr2	98
Qtr3	107
Qtr4	102

TRANSPOSE Procedure (교재 5.2.18)

: 행과 열의 전치

: 관측을 변수로, 변수를 관측으로 바꾸는 절차(procedure)

```
PROC TRANSPOSE DATA=input-data-set
                <OUT=output-data-set>
                <NAME=variable-name>
                <PREFIX=prefix>
                <SUFFIX=suffix> ;
    <BY <DESCENDING> variable-1 ......
        < <DESCENDING> variable-n > <NOTSORTED> ; >
    <VAR variable(s) ; >
    <ID variable ; >
RUN;
```

OUT=*output-data-set* 옵션 생략하면 원본 데이터에 덮어쓰지 않고 DATA#n(e.g. DATA1) 형태로 새로운 데이터셋으로 전치된 데이터 생성됨

Name 은 전치된 행의 값의 이름 지정.
디폴트는 _NAME_

PREFIX=*prefix* 는 전치된 변수명에 사용할 접두사 지정

SUFFIX=*suffix* 는 전치된 변수명에 사용할 접미사 지정
PREFIX, SUFFIX 생략시 COL*n* 형식(COL1, COL2, COL3, ...)

BY 문장에서는 by 그룹에서 사용할 기준 변수(들) 지정 (미리 정렬 필요)

VAR 문장에서는 전치시킬 대상 변수(들) 지정 또는 문자형 변수 전치
이 문장 생략하면 숫자형 변수만 전치됨(디폴트)

ID 문장에서는 새로운 변수명이 될 변수 값들을 지정

(예)
```
DATA original;
    INPUT x y z;
    DATALINES;
      12 19 14
      21 15 19
      33 27 82
      14 32 99
  RUN;
  PROC TRANSPOSE DATA=original OUT=transposed;
  RUN;
```

```
TITLE 'Original DATA';
PROC PRINT DATA=original;
RUN;
TITLE 'Transposed Data';
PROC PRINT DATA=Transposed;
RUN;
```

⇒

Original Data

OBS	x	y	z
1	12	19	14
2	21	15	19
3	33	27	82
4	14	32	99

Transposed Data

OBS	_NAME_	COL1	COL2	COL3	COL4
1	x	12	21	33	14
2	y	19	15	27	32
3	z	14	19	82	99

전치된 행의 값 이름은 디폴트로 **_NAME_** (NAME= 옵션 없으므로)
전치된 SAS 자료의 변수 이름은 디폴트로 **COL*n*** 형식의 이름 부여.
(ID 문장/suffix/prefix 없으므로)

(예)

```
DATA old;
   INPUT subject time x;
   DATALINES;
      1 2 12
      1 4 15
      1 7 19
      2 2 17
      2 7 14
      3 2 21
      3 4 15
      3 7 18
RUN;
PROC SORT DATA=Old;
     BY subject;
RUN;
PROC TRANSPOSE DATA=Old OUT=New PREFIX=value;
            /* PREFIX 는 새 열변수 이름들에 공통적인 접두사 지정 */
     BY subject;  /* BY 변수 subject을 기준으로 행과 열 전치 */
                  /* BY 변수 자체는 전치되지 않음 */
                  /* BY 변수에 의해 sorting 되어 있어야함 */
     ID time;  /* PREFIX 에서 지정한 접두사 value에 ID 문장에서
                  지정한 변수 time 값이 덧붙음 */
RUN;
PROC PRINT DATA=New;
RUN;
```

⇒

OBS	subject	time	x
1	1	2	12
2	1	4	15
3	1	7	19
4	2	2	17
5	2	7	14
6	3	2	21
7	3	4	15
8	3	7	18

<Old>

OBS	subject	_NAME_	value2	value4	value7
1	1	x	12	15	19
2	2	x	17	.	14
3	3	x	21	15	18

<New>

(예) DATA origin;

INPUT employee_id qtr1 qtr2 qtr3 qtr4 paid_by $ 20-40;

DATALINES;

120265 . . . 25 Cash or Check

120267 15 15 15 15 Payroll Deduction

120269 20 20 20 20 Payroll Deduction

120270 20 10 5 . Cash or Check

120271 20 20 20 20 Payroll Deduction

RUN;

PROC TRANSPOSE DATA=origin **OUT**=trans;

* VAR 문장 생략되면 숫자변수만 전치 (문자변수 paid_by는 전치 안됨);

RUN;

PROC PRINT DATA=trans NOOBS;

RUN;

⇒

OBS	employee_id	qtr1	qtr2	qtr3	qtr4	paid_by
1	120265	.	.	.	25	Cash or Check
2	120267	15	15	15	15	Payroll Deduction
3	120269	20	20	20	20	Payroll Deduction
4	120270	20	10	5	.	Cash or Check
5	120271	20	20	20	20	Payroll Deduction

<origin>

NAME	COL1	COL2	COL3	COL4	COL5
employee_id	120265	120267	120269	120270	120271
qtr1	.	15	20	20	20
qtr2	.	15	20	10	20
qtr3	.	15	20	5	20
qtr4	25	15	20	.	20

<trans>

paid_by 라는 문자변수는 제외되고 전치됨

[참고] 문자변수도 전치시키려면 VAR 문장 추가

(예) BY 문장에서 BY 변수 지정

PROC SORT DATA=origin;

BY employee_id;

RUN;

PROC TRANSPOSE DATA=origin **OUT**=trans2;

BY employee_id; /* BY 변수에 의해 sorting 되어 있어야함 */

```
RUN;
PROC PRINT DATA=trans2 NOOBS;
RUN;
```

⇒

employee_ id	_NAME_	COL1
120265	qtr1	.
120265	qtr2	.
120265	qtr3	.
120265	qtr4	25
120267	qtr1	15
120267	qtr2	15
120267	qtr3	15
120267	qtr4	15
120269	qtr1	20
120269	qtr2	20
120269	qtr3	20
120269	qtr4	20
120270	qtr1	20
120270	qtr2	10
120270	qtr3	5
120270	qtr4	.
120271	qtr1	20
120271	qtr2	20
120271	qtr3	20
120271	qtr4	20

BY 변수인 그룹변수 employee_id 이외의 모든 숫자변수에 대해 전치됨

(예) VAR 문장 추가하여 특정 변수 또는 문자변수 전치

```
PROC TRANSPOSE DATA=origin OUT=trans3;
   BY employee_id;
   VAR qtr1-qtr4;   /* VAR 문장에서 지정한 변수들만 전치 */
   * VAR qtr1-qtr4 paid_by;
RUN;
PROC PRINT DATA=trans3 NOOBS;
RUN;
```

⇒

employee_ id	_NAME_	COL1
120265	qtr1	.
120265	qtr2	.
120265	qtr3	.
120265	qtr4	25
120267	qtr1	15
120267	qtr2	15
120267	qtr3	15
120267	qtr4	15
120269	qtr1	20
120269	qtr2	20
120269	qtr3	20
120269	qtr4	20
120270	qtr1	20
120270	qtr2	10
120270	qtr3	5
120270	qtr4	.
120271	qtr1	20
120271	qtr2	20
120271	qtr3	20
120271	qtr4	20

(예) 변수명 변경 (OUT=에서 RENAME= 옵션으로 변수명 변경 가능)
(NAME= 옵션으로 디폴트인 _NAME_ 이름 변경)

```
PROC TRANSPOSE DATA=origin
               OUT=rotate2(RENAME=(col1=amount)) NAME=period;
     BY employee_id;
RUN;
PROC PRINT DATA=rotate2 NOOBS;  RUN;
```

⇒

employee_id	period	amount
120265	qtr1	.
120265	qtr2	.
120265	qtr3	.
120265	qtr4	25
120267	qtr1	15
120267	qtr2	15
120267	qtr3	15
120267	qtr4	15
120269	qtr1	20
120269	qtr2	20
120269	qtr3	20
120269	qtr4	20
120270	qtr1	20
120270	qtr2	10
120270	qtr3	5
120270	qtr4	.
120271	qtr1	20
120271	qtr2	20
120271	qtr3	20
120271	qtr4	20

```
PROC FREQ DATA=rotate2;
     TABLES period;
RUN;
```

⇒

FREQ 프로시저

NAME OF FORMER VARIABLE

period	빈도	백분율	누적 빈도	누적 백분율
qtr1	5	25.00	5	25.00
qtr2	5	25.00	10	50.00
qtr3	5	25.00	15	75.00
qtr4	5	25.00	20	100.00

NAME 이름을 변경시켰지만 _NAME_ 의 디폴트 label(NAME OF FORMER VARIABLE)은 그대로 유지

```
PROC FREQ DATA=rotate2;
     WHERE amount NE . ;
          /* amount 값이 결측이 아닌 것들 중에서 FREQ */
     TABLES period;
     LABEL period=' ';
RUN;
```

⇒

FREQ 프로시저

period	빈도	백분율	누적 빈도	누적 백분율
qtr1	4	25.00	4	25.00
qtr2	4	25.00	8	50.00
qtr3	4	25.00	12	75.00
qtr4	4	25.00	16	100.00

(예) ID 문장 지정

```
DATA orion;
    INPUT customer_id order_month sale_amount;
    LABEL customer_id='Customer ID';
    DATALINES;
      5  5  478.00
      5  6  126.80
      5  9   52.50
      5 12   33.80
     10  3   32.60
     10  4  250.80
     10  5   79.80
     10  6   12.20
     10  7  163.29
RUN;
PROC TRANSPOSE DATA=orion OUT=annual_orders;
RUN;
PROC PRINT DATA=annual_orders NOOBS;
RUN;
```

⇒

NAME	_LABEL_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9
customer_id	Customer ID	5	5.0	5.0	5.0	10.0	10.0	10.0	10.0	10.00
order_month		5	6.0	9.0	12.0	3.0	4.0	5.0	6.0	7.00
sale_amount		478	126.8	52.5	33.8	32.6	250.8	79.8	12.2	163.29

```
PROC TRANSPOSE DATA=orion OUT=annual_orders;
    BY customer_id;
    ID order_month; /* 변수값을 변수명으로 사용 */
                    /* order_month 는 숫자변수이므로 _변수값 */
                    /* PREFIX 등과 함께도 사용 가능 */
RUN;
```

⇒

customer_id	_NAME_	_5	_6	_9	_12	_3	_4	_7
5	sale_amount	478.0	126.8	52.5	33.8	.	.	.
10	sale_amount	79.8	12.2	.	.	32.6	250.8	163.29

7. 데이터 결합 (combining data)

- 데이터의 결합 (Merge 등)

8. Macro 언어

: 매크로 변수 설명 및 예제

- SAS 프로그램 내에서 매크로 변수 생성 및 사용
- 원하는 SAS code를 생성하기 위한 매크로 프로그램 작성 및 불러들여 사용

매크로 변수 type

- 자동 매크로 변수 (automatic macro variables)

: SAS 가 기동될 때 SAS 시스템으로부터 제공 받는 정보로, date, time 등

- 사용자 지정 매크로 변수 (user-defined macro variables)

: 사용자가 지정한 매크로 변수들

<자동 매크로 변수들 (automatic macro variables) 중 일부>

Name	Description
SYSDATE	SAS 기동 날짜 (DATE7.)
SYSDATE9	SAS 기동 날짜 (DATE9.)
SYSDAY	SAS 기동 요일
SYSTIME	SAS 기동 시간
SYSSCP	Operating System (OS, WIN, PH64, 등)
SYSVER	SAS version

매크로 변수 사용 방법은,

&*macro-variable-name*

(예)
```
DATA order_fact;
    INPUT customer_id @4 order_date DATE9.
          @14 delivery_date DATE9. order_id order_type product_id
          total_retail_price DOLLAR7.;
    FORMAT order_date DATE9. delivery_date DATE9.;
    DATALINES;
 63 11JAN2003 11JAN2003 1230058123 1 220101300017  $16.50
  5 15Jan2003 19JAN2003 1230080101 2 230100500026 $247.50
 45 20JAN2003 22JAN2003 1230106883 2 240600100080  $28.30
 41 28JAN2003 28JAN2003 1230147441 1 240600100010  $32.00
 ;
 RUN;
 FOOTNOTE1 "Created &systime &sysday, &sysdate9";
 FOOTNOTE2 "on the &sysscp System Using Release &sysver";
```

/* 문자열 안에서 매크로 변수 사용할 경우, 홑따옴표 말고 **겹따옴표** 사용해야 함 */

```
PROC PRINT DATA=order_fact NOOBS;
RUN;
```

⇒

SAS 시스템

customer_id	order_date	delivery_date	order_id	order_type	product_id	total_retail_price
63	11JAN2003	11JAN2003	1230058123	1	220101300017	16.5
5	15JAN2003	19JAN2003	1230080101	2	230100500026	247.5
45	20JAN2003	22JAN2003	1230106883	2	240600100080	28.3
41	28JAN2003	28JAN2003	1230147441	1	240600100010	32.0

Created 13:17 Friday, 30JAN2009
on the WIN System Using Release 9.2

%LET *variable=value* ;

: 사용자 지정 매크로 변수 생성

이미 존재하는 변수라면 값이 새로 지정된 값으로 대체됨
숫자값도 **문자값**으로 저장됨
대소문자 구분하여 저장됨
따옴표도 변수값의 일부로 인식됨
앞/뒤 공백(leading and trailing blanks)은 제거되고 변수값 저장됨
수식연산은 계산된 결과를 저장하는 것이 아닌 그대로의 값이 저장됨

(예) OPTIONS SYMBOLGEN;

```
/* SYMBOLGEN 옵션은 로그창에 매크로변수 값이 무엇인지 출력 */
%LET year=2006;
%LET city=Dallas, TX;
%LET fname=      Marie     ;
%LET name='     Marie Budson    ';
%LET total=10+2;
TITLE "&year, &city, &fname, &name, &total";
```

⟾ 로그창 내용 일부

```
---
SYMBOLGEN:  Macro variable YEAR resolves to 2006
SYMBOLGEN:  Macro variable CITY resolves to Dallas, TX
SYMBOLGEN:  Macro variable FNAME resolves to Marie
SYMBOLGEN:  Macro variable NAME resolves to '  Marie Hudson  '
SYMBOLGEN:  Macro variable TOTAL resolves to 10+2
```

%PUT _automatic_ ;

: 자동 매크로 변수에 대한 정보가 로그창에 출력

%PUT _user_ ;

: 사용자 지정 매크로 변수에 대한 정보가 로그창에 출력

%PUT _all_ ;

: 모든 매크로 변수에 대한 정보가 로그창에 출력

```
229  %PUT _all_;
GLOBAL YEAR 2006
GLOBAL TOTAL 10+2
GLOBAL NAME '  Marie Hudson  '
GLOBAL FNAME Marie
GLOBAL CITY Dallas, TX
AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCC 4
AUTOMATIC SYSCHARWIDTH 1
AUTOMATIC SYSCMD
AUTOMATIC SYSDATE 30JAN09
AUTOMATIC SYSDATE9 30JAN2009
AUTOMATIC SYSDAY Friday
AUTOMATIC SYSDEVIC
```

(예)
```
DATA one;
      DO x=-3 TO 3 BY 0.1;
            y=x**2;
            OUTPUT;
      END;
RUN;
PROC PLOT DATA=one;
      TITLE "Plot of Y=X**2";
      PLOT y*x;
RUN;
** ==> Macro 언어로;
%MACRO draw(lower=, upper=, incr=, func=);
DATA one;
      DO x=&lower TO &upper BY &incr;
            y=&func;
            OUTPUT;
      END;
```

```
RUN;
PROC PLOT DATA=one;
      TITLE "Plot of Y=&func";
      PLOT y*x;
RUN;
%MEND draw;    /* 또는 %MEND;  */
%draw(lower=-3, upper=3, incr=0.1, func=x**2);

/* y**2 에 대한 plot 이외의 다른 plot 도 가능 */
%draw(lower=0, upper=4, incr=0.05, func=exp(-x));
```

(예)

```
DATA one;
      INPUT y x @@;
      DATALINES;
      3 9 1 8 3 12
;
RUN;
DATA two;
      INPUT z x1 x2 @@;
      DATALINES;
      16 3 8 12 5 3 20 4 6 17 7 9
;
RUN;
```

```
%macro reg(dep=, indep=, dataset=);
      PROC REG DATA=&dataset;
            MODEL &dep=&indep;
      RUN;
      QUIT;
%mend reg;
%reg(dep=y, indep=x, dataset=one);
%reg(dep=z, indep=x1 x2, dataset=two);
```

⟹

```
PROC REG DATA=one;
    MODEL y=x;
RUN;
QUIT;
PROC REG DATA=two;
    MODEL z=x1 x2;
RUN;
QUIT;
```

(예) DATA sideways;
INPUT x1-x10 trt $;
DATALINES;
4 6 5 7 5 7 6 4 3 5 trt1
6 5 5 8 7 6 7 8 6 7 trt2
RUN;

```
DATA ttest;
        SET sideways;
        x=x1;   OUTPUT;
        x=x2;   OUTPUT;
        x=x3;   OUTPUT;
        x=x4;   OUTPUT;
        x=x5;   OUTPUT;
        x=x6;   OUTPUT;
        x=x7;   OUTPUT;
        x=x8;   OUTPUT;
        x=x9;   OUTPUT;
        x=x10;   OUTPUT;
        DROP x1-x10;
RUN;
```

PROC PRINT DATA=ttest;
RUN;

** ==> Macro 언어로; /* 연습용 SAS 프로그램 참고 */

```
%macro tr (num=);
        %DO j=1 %TO &num;
                x=x&j;
                OUTPUT;
        %END;
        DROP x1-x&num;
%mend tr;
DATA ttest;
        SET sideways;
        %tr (num=10);
RUN;
```

PROC TTEST DATA=ttest; /* independent two-sample t-test */
CLASS trt;

```
    VAR x;
RUN;
```

⇒

The TTEST Procedure

Variable: x

trt	N	Mean	Std Dev	Std Err	Minimum	Maximum
trt1	10	5.2000	1.3166	0.4163	3.0000	7.0000
trt2	10	6.5000	1.0801	0.3416	5.0000	8.0000
Diff (1-2)		-1.3000	1.2042	0.5385		

trt	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
trt1		5.2000	4.2582	6.1418	1.3166	0.9056	2.4035
trt2		6.5000	5.7273	7.2727	1.0801	0.7429	1.9719
Diff (1-2)	Pooled	-1.3000	-2.4314	-0.1686	1.2042	0.9099	1.7807
Diff (1-2)	Satterthwaite	-1.3000	-2.4345	-0.1655			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	18	-2.41	0.0266
Satterthwaite	Unequal	17.338	-2.41	0.0271

Equality of Variances

Method	Num DF	Den DF	F Value	Pr > F
Folded F	9	9	1.49	0.5648

(예)
```
DATA data1;
    INPUT y x @@;
    DATALINES;
    4 8 9 12 5 10
RUN;
DATA data2;
    INPUT z z1 z2 @@;
    DATALINES;
    12 7 3 15 8 3 17 5 2 16 4 2
RUN;

%macro reg (dataset=, dep=, indep=);
    TITLE1 "회귀분석";
    TITLE2 "반응변수: &dep,  설명변수: &indep";  /* 표제 붙이기 */
    PROC REG DATA=&dataset;
        MODEL &dep=&indep;
    RUN;
    QUIT;
%mend;
%reg (dataset=data1, dep=y, indep=x);
%reg (dataset=data2, dep=z, indep=z1 z2);
```

(예)
```
%macro simple (dset=, var=, stat=);
        PROC MEANS DATA=&dset &stat;
                TITLE1 "Proc Means with statistics : &stat";
                TITLE2 "Response Variable = &var";
                VAR &var;
        RUN;
    %mend simple;
    %simple (dset=data2, var=z2, stat=MEAN STD N);
    %simple (dset=data1, var=y, stat=MAX MIN N);
```

(예)
```
%macro subset (var1=, crit=, ds_in=, ds_out=, compare= );
        DATA &ds_out;
                SET &ds_in;
                IF &var1 &compare &crit THEN OUTPUT;
        RUN;
        TITLE "<데이터셋명: &ds_out>";
        PROC PRINT DATA=&ds_out;
        RUN;
    %mend subset;
    DATA one;
        INPUT x1 x2 x3 y @@;
        DATALINES;
        0.3 2 3 79 1.5 3 3 56 0.5 5 2 42 1.0 10 1 72
        0.8 4 1 52 0.3 5 3 74 0.5 9 3 30 1.2 4 2 35
        1.5 0 2 42 1.0 6 3 51 1.5 3 5 65 0.0 3 5 73
    ;
    RUN;
    %subset (var1=x1, crit=1, ds_in=one, ds_out=result1, compare=<);
    %subset (var1=y, crit=75, ds_in=one, ds_out=result2, compare=>=);
```

⇒

<데이터셋명: result1>

OBS	x1	x2	x3	y
1	0.3	2	3	79
2	0.5	5	2	42
3	0.8	4	1	52
4	0.3	5	3	74
5	0.5	9	3	30
6	0.0	3	5	73

<데이터셋명: result2>

OBS	x1	x2	x3	y
1	0.3	2	3	79

SET ..... (IN= .....)
추가

EMPLOYEE

name	age
Bruce	30
Dan	35

SALARY

name	salary
Bruce	40000
Bruce	35000
Dan	37000
Dan	.

The following SAS program is submitted:

```
data work.empsalary;
    merge work.employee (in = inemp)
                work.salary (in = insal);
    by name;
    if inemp and insal;
run;
```