Learning outcomes:

1. Rotating with DATA step using array and DO loop processing
2. Using TRANSPOSE procedure

SAS components learnt:

1. PROC TRANSPOSE
2. DO Loop

10.1 Rotating with DATA step

- A wide data set:
  - All information about one entity is stored in a single observation.
  - This data set structure is useful for data mining and generating report. For example, generate the total sales amount of a year. In SAS, we add one statement, TOTAL=SUM(of Qtr:).
  - An example: Five customers of Dorothy's supermarket are stored as follows.

Customer_ID	Qtr1	Qtr2	Qtr3	Qtr4	Paid_method
A123234	1234	5678	234		Credit card
B564646	3456	1234	2345	3316	Debit card
C900736	123				Cash
C845656	45	67		89	Octopus
D908765					Debit card

- A narrow data set:
  - All information about one entity is stored as multiple observations.
  - Missing value might be stored.
  - Dorothy's supermarket quarterly sales amount revisit:

Customer_ID	Period	Sales_amount
A123234	Qtr1	1234
A123234	Qtr2	5678
A123234	Qtr3	234
A123234	Qtr4	
B564646	Qtr1	3456
B564646	Qtr2	1234
B564646	Qtr3	2345
B564646	Qtr4	3316
C900736	Qtr1	123
C900736	Qtr2	
C900736	Qtr3	
C900736	Qtr4	
C845656	Qtr1	45
C845656	Qtr2	67
C845656	Qtr3	
C845656	Qtr4	89
D908765	Qtr1	
D908765	Qtr2	
D908765	Qtr3	
D908765	Qtr4	

- If the missing values are deleted, we can use PROC FREQ to count the number of customers who made purchase in Dorothy's supermarket in each period.

```
* p10_dorothy_supermarket.sas;
libname xlsdata
'd:\SAS_datasets\Dorothy_Supermarket_Sales.xlsx';
data consumers;
    set xlsdata.'Narrow_data_set$'n;
      if Sales_amount ne . ;
run;

proc print data=consumers;
run;

proc freq data=consumers;
table period /nocum;
run;
```

- The output:

```
        The FREQ Procedure

            Period

Period    Frequency     Percent
------------------------------
Qtr1              4       36.36
Qtr2              3       27.27
Qtr3              2       18.18
Qtr4              2       18.18
```

  - In first quarter, 4 customers had purchase. In second quarter 3 customers had purchase. In third quarter, 2 customers had purchase. In fourth quarter, 2 customers had purchase.

- Converting data from wide data set to narrow data set
  - Use of array to take the value of quarterly sales amount.
  - Use of DO loop for the 4 quarters
  - Use of OUTPUT statement in the DO loop
  - SAS program:

```
* p10_dorothy_supermarket.sas;
* rotate data;
proc print data=xlsdata.'Quarter_Sales$'n;
run;

data rotate (drop=qtr1-qtr4);
  array sales{4} qtr1-qtr4;
  set xlsdata.'Quarter_Sales$'n (drop=paid_method);
  do i = 1 to 4;
        if sales{i} ne . then do;
                Period=cats("Qtr",i);
                 Sales_amount=sales{i};
                 output;
               end;
      end;
run;

proc print data=rotate;
run;
```

- The output:

Obs	Customer_ ID	i	Period	Sales_ amount
1	A123234	1	Qtr1	1234
2	A123234	2	Qtr2	5678
3	A123234	3	Qtr3	234
4	B564646	1	Qtr1	3456
5	B564646	2	Qtr2	1234
6	B564646	3	Qtr3	2345
7	B564646	4	Qtr4	3316
8	C900736	1	Qtr1	123
9	C845656	1	Qtr1	45
10	C845656	2	Qtr2	67
11	C845656	4	Qtr4	89

- The data set has rotated.

Ex. 10.1: (a) Explain why "`drop=qtr1-qtr4`" is required for the rotation of wide data set to narrow data set. [Hint: Submit the program without the DROP option and print the result.]

(b) Why "`if sales{i} ne . then do;… end;`" is required ?

Ex. 10.2: (a) Write down the PDV information during the compilation stage for the rotate data portion of p10_dorothy_supermarket.sas.

Qtr1 N 8	Qtr2 N 8	Qtr3 N 8	Qtr4 N 8	Customer_ID $ 7	i N 8	Period $ 4	Sales_amount N 8

(b) Fill in the blanks of the following table for the PDV during execution.

Iteration	Qtr1 N 8	Qtr2 N 8	Qtr3 N 8	Qtr4 N 8	Customer_ID $ 7	i N 8	Period $ 4	Sales_amount N 8
1: DATA Step								
1: SET statement								
1: Do Statment								
1: IF-THEN Statment								
1: END statement								
1: IF-THEN Statment								
1: END statement								
1: IF-THEN Statment								
1: END statement								
1: END statement								

2: DATA Step								
2: SET statement								
2: Do Statment								
2: IF-THEN Statment								
2: END statement								
2: IF-THEN Statment								
2: END statement								
2: IF-THEN Statment								
2: END statement								
2: IF-THEN Statment								
2: END statement								
3: DATA Step								
3: SET statement	123	.	.	.	C900736	.		.

*highlighted text is output.

10.2 Using TRANSPOSE procedure

- General form
  - Syntax

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

  - *NAME* specifies a new name for the _NAME_ column. The value in this column identifies the variable that supplied the values in the row.
  - *BY* specifies the variable(s) to use to form BY groups. Note that sorting by BY-variable in PROC SORT is required unless NOTSORTED option is used.
  - The NOTSORTED option in the BY statement is used when the observations are not necessarily sorted in alphabetic or numeric order but grouped in another way, such as chronological order.
  - *VAR* specifies the variable(s) to transpose.
  - *ID* specifies the variable whose values will become the new variables.
  - *PREFIX* specifies a prefix for each new variable formed in the ID statement and replaces the underscore at the first position of the variable name.
- A procedure to re-structure a data set by transposing:
  - Selected variables into observations
  - Numeric variables by default
  - Character variables only if explicitly listed in a VAR statement
- Convert a wide data set into a narrow data set using PROC TRANSPOSE
  - Dorothy's supermarket customers data revisit
    - Using EXCEL to do the transpose option

Customer_ID	A123234	B564646	C900736	C845656	D908765
Qtr1	1234	3456	123	45	
Qtr2	5678	1234		67	
Qtr3	234	2345			
Qtr4		3316		89	
Paid_method	Credit card	Debit card	Cash	Octopus	Debit card

  - Using PROC TRANSPOSE:
    - Since only numeric values will be transposed, we drop the first letter of CUSTOMER_ID so that CUSTOMER_ID becomes a numeric field.

- Using EXCEL for doing transpose:

Customer_ID	123234	564646	900736	845656	908765
Qtr1	1234	3456	123	45	
Qtr2	5678	1234		67	
Qtr3	234	2345			
Qtr4		3316		89	

- The simplest PROC TRANSPOSE:

```
*p10_transpose_N.sas;
libname xlsdata 'd:\SAS_datasets\Dorothy_Supermarket_Sales.xlsx';
proc transpose data=xlsdata.'Quarter_Sales_N$'n out=transpose1;
run;

proc print data=transpose1;
run;
```

Obs	_NAME_	_LABEL_ *	COL1	COL2	COL3	COL4	COL5
1	Customer_ID	Customer_ID	123234	564646	900736	845656	908765
2	Qtr1	Qtr1	1234	3456	123	45	.
3	Qtr2	Qtr2	5678	1234	.	67	.
4	Qtr3	Qtr3	234	2345	.	.	.
5	Qtr4	Qtr4	.	3316	.	89	.

*_LABEL_ column was created because permanent labels were defined in the input data set. This column is **not included** in the output if there are **no** labels in the input data set.

- Add "BY CUSTOMER_ID NOTSORTED"

```
proc transpose data=xlsdata.'Quarter_Sales_N$'n out=transpose1by;
by CUSTOMER_ID NOTSORTED;
run;

proc print data=transpose1by;
run;
```

Obs	Customer_ ID	_NAME_	_LABEL_	COL1
1	123234	Qtr1	Qtr1	1234
2	123234	Qtr2	Qtr2	5678
3	123234	Qtr3	Qtr3	234
4	123234	Qtr4	Qtr4	.
5	564646	Qtr1	Qtr1	3456
6	564646	Qtr2	Qtr2	1234
7	564646	Qtr3	Qtr3	2345
8	564646	Qtr4	Qtr4	3316
9	845656	Qtr1	Qtr1	45
10	845656	Qtr2	Qtr2	67
11	845656	Qtr3	Qtr3	.
12	845656	Qtr4	Qtr4	89
13	900736	Qtr1	Qtr1	123
14	900736	Qtr2	Qtr2	.
15	900736	Qtr3	Qtr3	.
16	900736	Qtr4	Qtr4	.
17	908765	Qtr1	Qtr1	.
18	908765	Qtr2	Qtr2	.
19	908765	Qtr3	Qtr3	.
20	908765	Qtr4	Qtr4	.

Sorted by Customer_ID!

NOTSORTED option is applied to SAS data set ONLY!

Ex. 10.3: (a) Write a DATA step with temporary output SAS data set, 'Quarter_Sales' to store the data from worksheet 'Quarter_Sales' of d:\SAS_datasets\Dorothy_Supermarket_Sales.xlsx. No BY statement is used.
(b) Repeat the Procedure TRANSPOSE with NOTSORTED option in the BY statement.
(c) When BY statement is used in part (a), report any changes in output to (b).

Answers:

(a) & (b)

Outputs:

(c) The Customer ID column is sorted. Apart from this, no more things has been changed in the output.

- Add "VAR QTR1-QTR4": No effect, as QTR1-QTR4 are already variables.

```
proc transpose data=xlsdata.'Quarter_Sales_N$'n out=transpose1by;
by CUSTOMER_ID NOTSORTED;
var Qtr1-Qtr4;
run;

proc print data=transpose1by;
run;
```

Obs	Customer_ ID	_NAME_	_LABEL_	COL1
1	123234	Qtr1	Qtr1	1234
2	123234	Qtr2	Qtr2	5678
3	123234	Qtr3	Qtr3	234
4	123234	Qtr4	Qtr4	.
5	564646	Qtr1	Qtr1	3456
6	564646	Qtr2	Qtr2	1234
7	564646	Qtr3	Qtr3	2345
8	564646	Qtr4	Qtr4	3316
9	845656	Qtr1	Qtr1	45
10	845656	Qtr2	Qtr2	67
11	845656	Qtr3	Qtr3	.
12	845656	Qtr4	Qtr4	89
13	900736	Qtr1	Qtr1	123
14	900736	Qtr2	Qtr2	.
15	900736	Qtr3	Qtr3	.
16	900736	Qtr4	Qtr4	.
17	908765	Qtr1	Qtr1	.
18	908765	Qtr2	Qtr2	.
19	908765	Qtr3	Qtr3	.
20	908765	Qtr4	Qtr4	.

- Rename _NAME_ as period, COL1 as Sales_Amount and Drop the _LABEL_ column

```
proc transpose data=xlsdata.'Quarter_Sales_N$'n
out=transpose1byvar1(Rename=(COL1=Sales_Amount _NAME_=Period)
Drop=_LABEL_);
by CUSTOMER_ID;
var Qtr1-Qtr4;
run;

proc print data=transpose1byvar1;
run;
```

Obs	Customer_ ID	Period	Sales_ Amount
1	123234	Qtr1	1234
2	123234	Qtr2	5678
3	123234	Qtr3	234
4	123234	Qtr4	.
5	564646	Qtr1	3456
6	564646	Qtr2	1234
7	564646	Qtr3	2345
8	564646	Qtr4	3316
9	845656	Qtr1	45
10	845656	Qtr2	67
11	845656	Qtr3	.
12	845656	Qtr4	89
13	900736	Qtr1	123
14	900736	Qtr2	.
15	900736	Qtr3	.
16	900736	Qtr4	.
17	908765	Qtr1	.
18	908765	Qtr2	.
19	908765	Qtr3	.
20	908765	Qtr4	.

- Use "NAME= option" in the PROC TRANSPOSE statement and rename COL1 as Sales_Amount and Drop the _LABEL_ column, it produce the same output as above.

```
proc transpose data=xlsdata.'Quarter_Sales_N$'n
out=transpose1byvar1(Rename=(COL1=Sales_Amount) Drop=_LABEL_)
NAME=Period;
by CUSTOMER_ID;
var Qtr1-Qtr4;
run;

proc print data=transpose1byvar1;
run;
```

Ex. 10.4: If the CUSTOMER_ID is kept with first character letter, rewrite a PROC TRANSPOSE to obtain the transpose of the data set and list the output. The data are stored in the worksheet 'Quarter_sales' of Dorothy_supermarket_sales.xlsx.

Answers:

- In order to remove the missing values, WHERE= option for data set is used.

```
SAS-data-set(WHERE=(where-expression))
```

This WHERE option is applied to the output data set given in the OUT= option.

```
proc transpose data=xlsdata.'Quarter_Sales_N$'n
out=transpose1byvar1(Rename=(COL1=Sales_Amount)Drop=_LABEL_
WHERE=(Sales_Amount ne .)) NAME=Period;
by CUSTOMER_ID NOTSORTED;
var Qtr1-Qtr4;
run;

proc print data=transpose1byvar1;
run;
```

Obs	Customer_ ID	Period	Sales_ Amount
1	123234	Qtr1	1234
2	123234	Qtr2	5678
3	123234	Qtr3	234
4	564646	Qtr1	3456
5	564646	Qtr2	1234
6	564646	Qtr3	2345
7	564646	Qtr4	3316
8	845656	Qtr1	45
9	845656	Qtr2	67
10	845656	Qtr4	89
11	900736	Qtr1	123

- Using PROC FREQ to obtain the number of customers who purchased in Dorothy's supermarket in each quarter.

```
proc freq data=transpose3;
table period/ nocum nopct;
run;
```

Period	Frequency
Qtr1	4
Qtr2	3
Qtr3	2
Qtr4	2

- Converting a narrow data set to a wide data set
  - Here is the home delivery order of Dorothy's supermarket:

Customer_ID	Delivery_order_month	Sales_Amount
2	1	500
2	2	568
2	3	789
3	3	234
3	4	567
3	5	987
4	1	789
4	5	546
4	6	2745

- Using the simplest PROC TRANSPOSE

```
*p10_TRANSPOSE_N.sas;
*conversion of narrow data set to wide data set;
libname xls 'D:\SAS_datasets\Dorothy_supermarket_month.xlsx';
proc transpose data=xls.'Home_Delivery$'n out=wide1;
run;

proc print data=wide1;
run;
```

Obs	_NAME_	_LABEL_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9
1	Customer_ID	Customer_ID	2	2	2	3	3	3	4	4	4
2	Delivery_order_month	Delivery_order_month	1	2	3	3	4	5	1	5	6
3	Sales_Amount	Sales_Amount	500	568	789	234	567	987	789	546	2745

- CUSTOMER_ID is a group variable. Therefore use “BY CUSTOMER_ID”

```
*p10_TRANSPOSE_N.sas;
*conversion of narrow data set to wide data set;
libname xls 'D:\SAS_datasets\Dorothy_supermarket_month.xlsx';
proc transpose data=xls.'Home_Delivery$'n out=wide2;
BY CUSTOMER_ID;
run;

proc print data=wide2;
run;
```

Obs	Customer_ ID	_NAME_	_LABEL_	COL1	COL2	COL3
1	2	Delivery_order_month	Delivery_order_month	3	2	1
2	2	Sales_Amount	Sales_Amount	789	568	500
3	3	Delivery_order_month	Delivery_order_month	5	4	3
4	3	Sales_Amount	Sales_Amount	987	567	234
5	4	Delivery_order_month	Delivery_order_month	6	5	1
6	4	Sales_Amount	Sales_Amount	2745	546	789

- Change the Delivery_order_month to variable, as the wide form should have month as variables. Use ID statement:

```
*p10_TRANSPOSE_N.sas;
*conversion of narrow data set to wide data set;
libname xls 'D:\SAS_datasets\Dorothy_supermarket_month.xlsx';
proc transpose data=xls.'Home_Delivery$'n out=wide3;
by customer_id;
id delivery_order_month;
run;

proc print data=wide3;
run;
```

Obs	Customer_ ID	_NAME_	_LABEL_	_3	_2	_1	_5	_4	_6
1	2	Sales_Amount	Sales_Amount	789	568	500	.	.	.
2	3	Sales_Amount	Sales_Amount	234	.	.	987	567	.
3	4	Sales_Amount	Sales_Amount	.	.	789	546	.	2745

- The delivery_oder_month variables were 1, 2, 3, 4, 5, 6 but 1 becomes _1, 2 becomes _2, 3 becomes _3, etc.
- Another observation is that the month variables are not in order.

- Use Prefix to give a prefix to ID:

```
*p10_TRANSPOSE_N.sas;
*conversion of narrow data set to wide data set;
libname xls 'D:\SAS_datasets\Dorothy_supermarket_month.xlsx';
proc transpose data=xls.'Home_Delivery$'n out=wide4 prefix=month;
by customer_id;
id delivery_order_month;
run;

proc print data=wide4;
run;
```

Obs	Customer_ ID	_NAME_	_LABEL_	month3	month2	month1	month5	month4	month6
1	2	Sales_Amount	Sales_Amount	789	568	500	.	.	.
2	3	Sales_Amount	Sales_Amount	234	.	.	987	567	.
3	4	Sales_Amount	Sales_Amount	.	.	789	546	.	2745

- To put the months in order, use RETAIN statement in data step:

```
*p10_TRANSPOSE_N.sas;
*conversion of narrow data set to wide data set;
data wide4_order;
retain customer_ID month1-month6;
  set wide4;
  drop _name_ _LABEL_;
run;

proc print data=wide4_order;
run;
```

Obs	customer_ ID	month1	month2	month3	month4	month5	month6
1	2	500	568	789	.	.	.
2	3	.	.	234	567	987	.
3	4	789	.	.	.	546	2745

- RETAIN statement
  - Change the order of the variables in an existing data set permanently.
  - The placement of the RETAIN statement should be BEFORE the set statement so that PDV is initialized before reading an input data set
  - If new variables are also listed in the RETAIN statement, their values are retained from one iteration to the next. Normally new variables are reinitialized on every iteration. Retaining their values might generate undesirable results.

- Examine the ordering of the data set: USE PROC CONTENTS

```
*p10_TRANSPOSE_N.sas;
*conversion of narrow data set to wide data set;
proc contents data=wide4_order;
run;
```

```
The CONTENTS Procedure

Data Set Name        WORK.WIDE4_ORDER                      Observations          3
Member Type          DATA                                  Variables             7
Engine               V9                                    Indexes               0
Created              Tuesday, August 21, 2012 09:05:09 PM  Observation Length    56
Last Modified        Tuesday, August 21, 2012 09:05:09 PM  Deleted Observations  0
Protection                                                 Compressed            NO
Data Set Type                                              Sorted                NO
Label
Data Representation  WINDOWS_32
Encoding             wlatin1  Western (Windows)


                              Engine/Host Dependent Information

Data Set Page Size          8192
Number of Data Set Pages    1
First Data Page             1
Max Obs per Page            145
Obs in First Data Page      3
Number of Data Set Repairs  0
Filename                    C:\Users\Dorothy\AppData\Local\Temp\SAS Temporary
                            Files\_TD2408\wide4_order.sas7bdat
Release Created             9.0202M3
Host Created                W32_VSHOME


  Alphabetic List of Variables and Attributes

#    Variable       Type    Len    Label

1    customer_ID    Num       8    Customer_ID
2    month1         Num       8
3    month2         Num       8
4    month3         Num       8
5    month4         Num       8
6    month5         Num       8
7    month6         Num       8
```

- Advantage of each restructuring method
  - The TRANSPOSE procedure
    - It might eliminate the need for a complex DATA step
    - It requires very little code to restructure data
  - The DATA step
    - It can create multiple data sets
    - It can direct output to data sets based on data set contributors
    - It enables FIRST. and LAST. Processing
    - It enables complex data manipulation