


Learning outcomes:

1. Identify data errors
2. Identify procedure for validating data
3. Identify techniques for cleaning data
4. Manipulate SAS data sets
5. Create variable in SAS data set
6. Use of SAS variable list
7. Converting data type

SAS components learnt:

1. PROC APPEND
2. PROC FREQ
3. PROC MEANS
4. PROC UNIVARIATE
5. WHERE statement
6. IF-THEN/ELSE statement
7. Arithmetic Operators
8. SAS built-in date, characters and numeric functions
9. Rename option in SET statement
10. PUT and INPUT functions

4.1 Identify data errors

 Data errors occur when

- data is not correctly collected. Logical problems may be found;
- typos occurs during data entry;
- data values are not appropriate for the SAS statements that are specified in a program, for example, data type is numeric but coded as characters in data sets.

 **Ex. 4.1:** Circle any potential errors of the partial data set of AMA graduates in 1993

Uni_ID	First_Name	Last_Name	Gender	DOB	Current_Salary	Job_Title	Country	Phone	Join_Date
90010101	Peter	Smith	M	1/7/1965	600000	Professor	Austria	(00431)58000	7/9/1997
89241253	Zhian	Yang	M	15/8/1964	750000	Associate Professor	macau	(00853)123456	1/9/1994
91021201	Chi Wah	Chan	F	6/8/1973	500000	Consultant	Hong Kong	(00852)25698561	1/8/1993
89012345	Amy	Yip	W	5/6/1972	1000000	Government statistician	Hong KONG	(00852)29090612	1/8/1938
90893208	Tony	Brown	M	8/6/1972	555000	Lecturer	Switzerland	(004441)1234567	7/9/1997
90893208	Tony	Chung	M	9/9/1974	34567	Statistical Officer	HONG KONG	(00852)37090612	unknown
92304567	Rebecca	Wicker	G	8/6/1972	-70000		Hong Kong	123	79/9/1997

Answers:

 Data requirements

- The following criteria are commonly used to set up the rules for validating or cleaning the data:
 - Data type – numeric or characters
 - Valid range and uniqueness of numeric data
 - Valid forms of character string

- Logic between variables in the data
- Any missing values

- ✚ When SAS encounters a data error,
 - in the SAS log:
 - a note that describes the error is printed
 - the input record (contents of the input buffer) being read is displayed
 - the values in the SAS observation (contents of the PDV) being created are displayed
 - a missing value is assigned to the appropriate SAS variable
 - execution continues

- ✚ Data are required to be cleaned before use. Data requirements are defined for potential invalid data error detection and these errors can be detected using procedure steps, e.g. PROC UNIVARIATE, MEANS, FREQ and PRINT.

- ✚ During the execution of DATA steps for importing data from a delimited file, SAS creates automatically two variables:

SAS variables	Values	meaning
<code>_N_</code>	Natural numbers, i.e. 1, 2, 3, ...	Count the number of time DATA step iterates
<code>_ERROR_</code>	- 0 - 1	0 – no errors; 1 – at least one error(s) detected

4.2 Validating data

- ✚ As mentioned in previous section, “`_N_`” and “`_ERROR_`” are created automatically during the execution of DATA step. To handle the invalid data:

Interface	Techniques
Log window	Interpret the error message. Note that the value of variable <code>_N_</code> (say <i>n</i>) represents the <i>n</i> -th observation having errors.
SAS data set	<p>Output the data portion with <code>_ERROR_=1</code> into a new data set for detail examination. A portion of the sample program:</p> <pre> DATA DATA_ERR DATA_OK; Length ...; Infile source-data-csv-file dlm=','; Input ...; If _ERROR_=1 then output DATA_ERR; ELSE output DATA_OK; RUN; </pre> <p>← Implicit RETURN;</p>

- IF-THEN-ELSE statement is used for the multiple data set outputs:

```
IF expression THEN statement1;
ELSE statement2;
```

- Expression is a sequence of operands and operators which form a condition
- Statement 1 will be executed when the condition in the expression is satisfied.
- Statement 2 will be executed when the condition in the expression is not satisfied.

- Explicit Output statement is used. It syntax:

```
OUTPUT <libref.>data-set-name;
```

- Libref is used only when data will be output permanently. Note that LIBNAME statement is required to define Libref before the OUTPUT statement.
- The data in above format is stored as 'AMAggrad93.csv' and a SAS program is used to read the data. For the portion of data with error detected by SAS, it is read into a temporary data set INV_DATA and the remaining part is dumped into another temporary data V_DATA.

```
* p4example1.sas - data type error detection;
data INV_DATA V_DATA;
  infile 'd:\SAS_datasets\AMAggrad93.csv' dlm=',';
  length First_Name $ 20 Last_Name $20 Gender $1
         Job_Title $25 Country $25 Phone $ 20;
  input Uni_ID First_Name $ Last_Name $ Gender $ DOB:DDMMYY.
        Current_Salary: Dollar. Job_Title $ Country $ Phone $
        Join_Date :DDMMYY.;
  if _ERROR_=1 then output INV_DATA;
  ELSE output V_DATA;
run;
```

Partial SAS Log:

```
12 *p4example1.sas;
13 data INV_DATA V_DATA;
14 infile 'd:\SAS_datasets\AMAggrad93.csv' dlm=',';
15 length First_Name $ 20 Last_Name $20 Gender $1
16        Job_Title $25 Country $25 Phone $ 20 ;
17 input Uni_ID First_Name $ Last_Name $ Gender $ DOB:DDMMYY.
18        Current_Salary: Dollar. Job_Title $ Country $ Phone $ Join_Date:DDMMYY.;
19 if _ERROR_=1 then output INV_DATA;
20 ELSE output V_DATA;
21 run;
```

NOTE: The infile 'd:\SAS_datasets\AMAggrad93.csv' is:
Filename=d:\SAS_datasets\AMAggrad93.csv,
RECFM=V,LRECL=256,File Size (bytes)=599,
Last Modified=13Jul2012:15:38:52,
Create Time=13Jul2012:12:00:31

NOTE: Invalid data for Join_Date in line 6 84-90.
RULE: -----1-----2-----3-----4-----5-----6-----7-----8-----
6 90893208,Tony,Chung,M,9/9/1974,34567,Statistical Officer,HONG KONG,(00852)37090612,unkno
89 wn 90
First_Name=Tony Last_Name=Chung Gender=M Job_Title=Statistical Officer Country=HONG KONG
Phone=(00852)37090612 Uni_ID=90893208 DOB=5365 Current_Salary=34567 Join_Date=. _ERROR_=1 _N_=6
NOTE: Invalid data for Join_Date in line 7 60-68.
7 92304567,Rebecca,Wicker,G,8/6/1972,-70000, ,Hong Kong,123 ,79/9/1997 68
First_Name=Rebecca Last_Name=Wicker Gender=G Job_Title= Country=Hong Kong Phone=123
Uni_ID=92304567 DOB=4542 Current_Salary=-70000 Join_Date=. _ERROR_=1 _N_=7
NOTE: 7 records were read from the infile 'd:\SAS_datasets\AMAggrad93.csv'.
The minimum record length was 68.
The maximum record length was 94.
NOTE: The data set WORK.INV_DATA has 2 observations and 10 variables.
NOTE: The data set WORK.V_DATA has 5 observations and 10 variables.
NOTE: DATA statement used (Total process time):
real time 0.03 seconds
cpu time 0.03 seconds



Ex. 4.2: Read the above partial SAS log and answer the following questions:

- (a) How many observations are output to INV_DATA?
- (b) How many observations are output to V_DATA?
- (c) Write down the full name of the observations which are identified as error data by SAS.

- Outputs of the INV_DATA from PROC PRINT:

The SAS System							15:40 Friday, July 13, 2012			1	
O b s		F i r s t N a m e	L a s t N a m e	G e n d e r	J o b T i t l e	C o u n t r y	P h o n e	U n i D	D O B	C u r r e n t S a l a r y	J o i n t D a t e
1	Tony	Chung	M	Statistical Officer	HONG KONG	(00852)37090612	90893208	5365	34567	.	
2	Rebecca	Wicker	G		Hong Kong	123	92304567	4542	-70000	.	

- The above exercise shows only invalid date and wrong data type problems are identified. The data for these observations are changed to missing (i.e. a period) in the SAS data set. Note that ranges, uniqueness problem, logical errors of the data cannot be identified using these two methods.

🚩 Procedures UNIVARIATE, MEANS, FREQ and PRINT are more powerful for the data error detection:

Procedure	Statements used	Invalid data type detected	Error detection
FREQ	<ul style="list-style-type: none"> - TABLE - NLEVELS 	Character/ Numeric	<ul style="list-style-type: none"> - Invalid category / categories - Missing data
UNIVARIATE	<ul style="list-style-type: none"> - VAR 	Numeric	<ul style="list-style-type: none"> - Extreme observations (as the highest and lowest values output with corresponding observation number)
MEANS	<ul style="list-style-type: none"> - VAR - N - NMISS - MAX - MIN 	Numeric	<ul style="list-style-type: none"> - Extreme observations (as minimum and maximum output in the summary) - Missing data
PRINT	<ul style="list-style-type: none"> - VAR - WHERE 	Character / Numeric	<ul style="list-style-type: none"> - Subsetting the data with any invalid character string or numeric values

- We combine the two files into one main file for error checking by Procedure APPEND:

```
PROC APPEND BASE=Base-SAS-data-set DATA=SAS-data-set;  
RUN;
```

- The SAS-data-set will be added to the Base-SAS-data-set.

```
* p4example1.sas - combine valid and invalid data;
PROC APPEND BASE=V_DATA DATA=INV_DATA;
RUN;
```

- The observations of data set, WORK.INV_DATA are copied into the temporary data set WORK.V_DATA. The data set WORK.V_DATA contains all observations for data cleaning.
- PROC FREQ:
 - Produce one-way to n -way frequency tables.
 - Syntax

```
PROC FREQ DATA=SAS-data-set <NLEVELS>;
TABLE variable(s)<_ALL_ /NOPRINT>;
RUN;
```

- NLEVELS option displays a table which provides the number of distinct values for each variable name ;
- TABLE statement: “*” is used for two-way to n -way frequency tables. E.g TABLE statement for a two-way table is “TABLE variable1 * variable2”, three-way table is “TABLE variable1 * variable2 * variable3”. ‘_ALL_’ option is used to display all variables.
- NOPRINT is used when frequency counts are not required.

- Use to check the categorical data. For example, gender and country and the distribution.

```
* p4example1.sas - freq;
Proc FREQ DATA=V_DATA;
table gender country job_title;
run;
```

- The output:

The SAS System			15:40 Friday, July 13, 2012		2
The FREQ Procedure					
Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
F	1	14.29	1	14.29	
G	1	14.29	2	28.57	
M	4	57.14	6	85.71	
W	1	14.29	7	100.00	

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
Austria	1	14.29	1	14.29	
HONG KONG	1	14.29	2	28.57	
Hong Kong	1	14.29	3	42.86	
Hong Kong	2	28.57	5	71.43	
Switzerland	1	14.29	6	85.71	
macau	1	14.29	7	100.00	

Job_Title	Frequency	Percent	Cumulative Frequency	Cumulative Percent	
Associate Professor	1	16.67	1	16.67	
Consultant	1	16.67	2	33.33	
Government statistician	1	16.67	3	50.00	
Lecturer	1	16.67	4	66.67	
Professor	1	16.67	5	83.33	
Statistical Officer	1	16.67	6	100.00	
Frequency Missing = 1					

(ii) When NLEVELS option is added, the following table is added into the output:

Number of Variable Levels			
Variable	Levels	Missing Levels	Nonmissing Levels
Gender	4	0	4
Country	6	0	6
Job_Title	7	1	6

Ex. 4.3: Read the above SAS output and fill in the blanks of the following tables:

Variable	Any problems?	Any missing observations?
Gender		
Country		
Job Title		

(d) Use PROC PRINT to check the whole or partial record of the invalid data. For example, which observations do not have valid gender code.

```
* p4example1.sas - freq;
PROC PRINT DATA=V_DATA;
    WHERE GENDER NE 'F' and GENDER NE 'M';
RUN;
```

And the output:

F i r s t N a m e	L a s t N a m e	G e n d e r	J o b T i t l e	C o u n t r y	P h o n e	...
4 Amy	Yip	W	Government statistician	Hong Kong	(00852)29090612	...
7 Rebecca	Wicker	G		Hong Kong	123	...

Ex. 4.4: Write a SAS program to find whom has/have missing job title.

--

Answer:

- PROC UNIVARIATE:
 - (a) Produce reports of moments, basic statistical measures, tests for locations, quantiles, extreme observations, missing values. Note that extreme observations and missing values are useful in the data validation.
 - (b) Syntax

```
PROC UNIVARIATE DATA=SAS-data-set <NEXTROBS=n>;
VAR variable(s);
RUN;
```

Nextrobs=*n* indicates displaying *n*-th smallest and largest observation. The default value is 5. NEXTROBS=0 suppresses the table of extreme observations.

- (c) Used to check numerical variables. For example, current_salary.

```
*p4example1.sas - univariate;
PROC UNIVARIATE DATA=V_DATA;
var Current_Salary ;
RUN;
```

The output gives the moments, quantiles and extreme observations:

The UNIVARIATE Procedure			
Variable: Current_Salary			
Moments			
N	7	Sum Weights	7
Mean	481366.714	Sum Observations	3369567
Std Deviation	379609.993	Variance	1.44104E11
Skewness	-0.4088417	Kurtosis	-0.6794406
Uncorrected SS	2.48662E12	Corrected SS	8.64622E11
Coeff Variation	78.860873	Std Error Mean	143479.091
Basic Statistical Measures			
Location		Variability	
Mean	481366.7	Std Deviation	379610
Median	555000.0	Variance	1.44104E11
Mode	.	Range	1070000
		Interquartile Range	715433
Tests for Location: Mu0=0			
Test	-Statistic-	-----p Value-----	
Student's t	t 3.354961	Pr > t	0.0153
Sign	M 2.5	Pr >= M	0.1250
Signed Rank	S 12	Pr >= S	0.0469

Quantiles (Definition 5)	
Quantile	Estimate
100% Max	1000000
99%	1000000
95%	1000000
90%	1000000
75% Q3	750000
50% Median	555000
25% Q1	34567
10%	-70000

NEXTROBS=0 suppresses the following table.

The UNIVARIATE Procedure			
Variable: Current_Salary			
Extreme Observations			
-----Lowest-----		-----Highest-----	
Value	Obs	Value	Obs
-70000	7	500000	3
34567	6	555000	5
500000	3	600000	1
555000	5	750000	2
600000	1	1000000	4

The quantiles and the extreme observations are most useful for error detection.

- (d) Use PROC PRINT for identifying the observations with potential error. In our case `current_salary < 0` and `current_salary=34567` is required to be checked as it is 10 times less than the other `current_salary` of other observations.

```
PROC PRINT DATA=V_DATA; /*p4example1.sas - univariate*/  
var first_name last_name Current_Salary;  
where Current_Salary < 0 or Current_Salary=34567;  
run;
```

Output:

Obs	First_ Name	Last_ Name	Current_ Salary
6	Tony	Chung	34567
7	Rebecca	Wicker	-70000

- PROC MEANS:
 - (a) Produce a basic statistics report of the data, for example number of non-missing values, mean, standard deviation, etc.
 - (b) Syntax

```
PROC MEANS DATA=SAS-data-set <statistics>;  
VAR variable(s);  
RUN;
```

Statistics refer the options N, NMISS, MIN, MAX, MEAN, STDDEV, by default. N- number of non-missing values, NMISS- number of missing values, MEAN – average, STDDEV – standard deviation. Specify the statistic if less statistics are required.

- (c) Used to check numerical variables. For example, `current_salary`.

```
PROC MEANS DATA=V_DATA; /*p4example1.sas - means*/  
var Current_Salary ;  
RUN;
```

- (d) The output

The MEANS Procedure				
Analysis Variable : Current_Salary				
N	Mean	Std Dev	Minimum	Maximum
7	481366.71	379609.99	-70000.00	1000000.00

Extreme values can be checked. PROC PRINT is used for invalid observation identification.

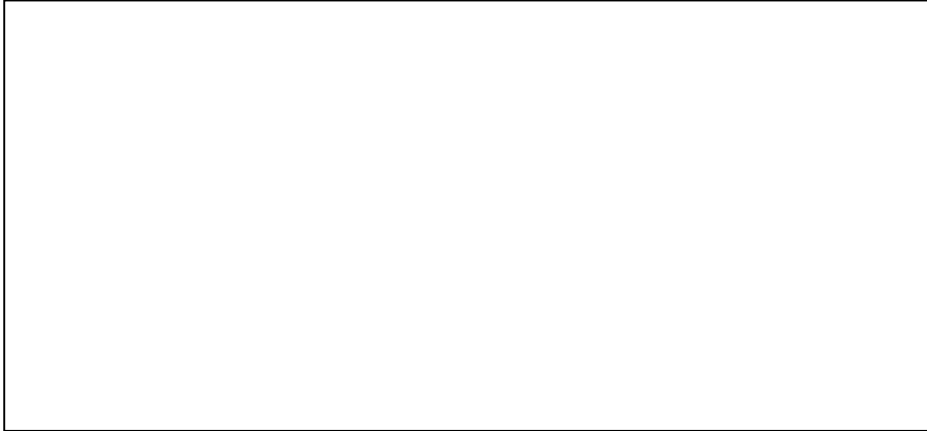
- PROC PRINT:
 - (a) The data requirement can be written in the WHERE statement so that observations with invalid data can be printed out. For example, to print out the observations with gender not equal to 'F' or 'M' or salary is negative or missing data for these two variables: WHERE (GENDER NE 'F' and GENDER NE 'M') or GENDER=' ' or Current_Salary='.' or Current_Salary < 0.
 - (b) Used to identifying the observations with errors.

 **Ex. 4.5:** Suggest a method to check the uniqueness of `Uni_ID`.

Answer:



The output:



- ✚ For checking more complicated data within one variable, other techniques like extracting a string from variables of observations, conversion of data from character to numeric may be required.
- ✚ For logic checking, IF-THEN statement is normally required. In the example the variable Join_Date represents the date for joining the organization for the job. So Join_Date should be larger than DOB (Date of Birth).

```
/*p4example1.sas - check logic*/  
DATA wrong_logic OK_logic;  
  set V_DATA;  
  If DOB>Join_Date then output wrong_logic;  
  else output OK_logic;  
run;  
  
Proc Print data=wrong_logic;  
  var first_name last_name DOB Join_date;  
  format DOB Join_Date DDMMYY9.;  
run;
```

The output:

Obs	First_ Name	Last_ Name	DOB	Join_Date
1	Amy	Yip	05/06/72	01/08/38
2	Tony	Chung	09/09/74	.
3	Rebecca	Wicker	08/06/72	.

The output shows that observations 1, 2 and 3 have logical errors.

- ✚ More techniques will be discussed in later sections.

4.3 Cleaning data

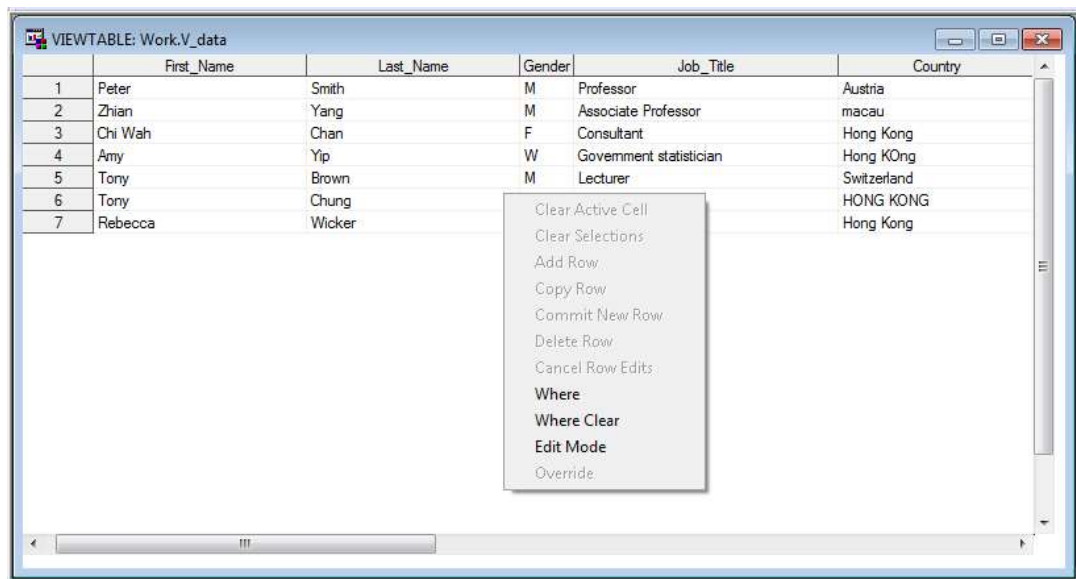
When the invalid data are identified, they would be cleaned into valid data.

Techniques for cleaning data:

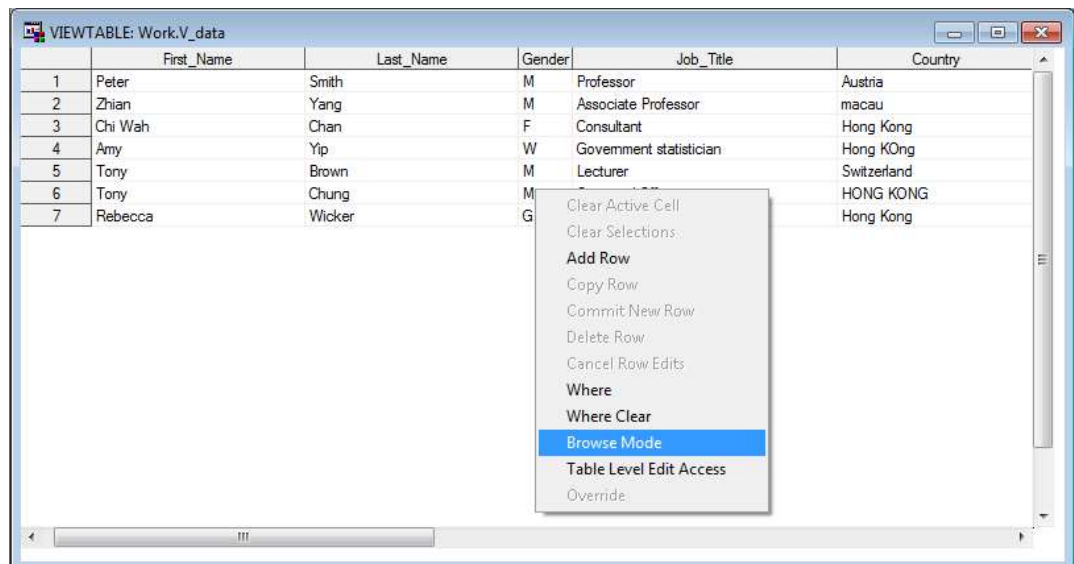
- Editing raw data file outside of SAS
- **Interactively editing data set using VIEWTABLE**
- **Programmatically editing data set using the DATA step**
- Programmatically editing data set using the SQL procedure
- Using the SAS DataFlux product of Power Studio

VIEWTABLE

- Double click “Libraries” > double click the SAS data set for editing
- Click on the “Viewtable” window and then right click to change to “Edit Mode” for editing



- Click on “Browse Mode” after finishing the edit.



✚ Programmatically editing data set using the DATA step

- Create a new data set using DATA step together with assignment statement and IF-THEN/ELSE statement.
- Assignment statement:

```
variable = expression;
```

- variable names an existing or new variable.
- expression is a sequence of operands and operators that form a set of instructions that produce a value.
- Operands are
 - (a) character constants, e.g. 'F'
 - (b) numeric constants, e.g. 2690
 - (c) date constants, e.g. '11Sep2011'd
 - (d) character variables
 - (e) numeric variables
- Operators are
 - (a) symbols that represent an arithmetic calculation, +, -, ×, /
 - (b) SAS functions, UPCASE(country)

- IF-THEN/ELSE statement

```
IF expression THEN an executable statement;
```

```
IF expression THEN statement 1;  
ELSE statement 2;
```

- Expression is a sequence of operands and operators that form a set of instructions that define a **condition** for selecting observations
 - Statement 1 is any executable statement which is executed when the expression is true; otherwise statement 2 will be executed.
- AMAGrad93 data set revisit:
 - The correction of data is shown below.

Variable	Invalid data	Corrections
Gender	'W', 'G'	'W' changed to 'F' 'G' changed to 'F'
Country	HONG KONG, Hong KONG, macau	Change the code with proper case. Use PROPCASE().
Join_Date	1/8/1938 or missing	1/8/1938 changed to 1/8/1998 Missing: Observation 6: 17/7/1998 Observation 7: 9/9/1997
Job Title	missing	Changed to 'Programmer'
Current Salary	-70000	Changed to 20000 (Observation 7)

- DO Statement:

```
DO;  
Statement-1;  
!  
Statement-n;  
END;
```

- MDY function returns a SAS date value from numeric month, day and year values.
- Use DATA step to create a new data set for cleaned data. The data set includes all the changes above and then use PROC PRINT to print the cleaned data. IF-THEN and DO-END statements are used.

```
DATA C_DATA; /*p4_clean_data.sas */
  set Work.V_DATA;
  If gender='W' then gender='F';
  If gender='G' then gender='F';
  country=Propcase(country);
  If Join_date=MDY(8,1,1938) then Join_date=MDY(8,1,1998);
  If _N_=6 then Join_Date=MDY(7,17,1998);
  If _N_= 7 then do;
    Current_Salary=20000;
    Job_title='Programmer';
    Join_Date=MDY(9,9,1997);
  end;

run;

proc print data=c_data;
format DOB Join_date ddmmyy.;
run;
```

4.4 Manipulating data

- Create SAS variables in DATA step
 - Using assignment statement (discussed in Chapter 3)
 - Expression of the assignment statement can be formed by
 - Arithmetic operators

Symbol	Definition	Priority	Example
**	Exponentiation	I	S= a**b;
-	Negative prefix	I	S= -a;
*	Multiplication	II	P= a*b;
/	Division	II	Ratio= a/b;
+	Addition	III	Y= a+b;
-	Subtraction	III	X= a-b;

Note: If **a missing value** is an operand for an arithmetic operator, the result is a missing value.



Ex. 4.6: Calculate the following:

Expression	Value
7+3**2/4	
(7+3)**2/4	
(7+3)**(2/4)	

- SAS functions

```
Function-name(argument1,argument2,...);
```

- The number of arguments depends on function.
- Arguments are separated with commas.
- Commonly used date functions:

Date Functions	SAS action	Examples
TODAY()	Return the current date as a SAS date value	Sign_date=Today();
MDY(month, day, year)	Return a SAS date value from numeric month, day and year values. <ul style="list-style-type: none"> Month: 1-12 Day: 1-31 Year: four digits 	Join_date=MDY(7,31,1997); Bonus_date=mdy(month(Join_date),15,2010);
YEAR(SAS-date)	Return a four-digit value for year of a SAS date	Sign_Year=Year(Today()); Sign_Year=Year(Sign_date); Sign_Year=Year('06JUL12'd);
QTR(SAS-date)	Return the quarter number (1, 2, 3, 4) of the quarter for a SAS date	Sign_Quarter= QTR(Sign_date);
MONTH(SAS-date)	Return the month number (1-12) of the month for a SAS date.	Sign_month=Month(Sign_date);
DAY(SAS-date)	Return the day number (1-31) of the day for a SAS date.	Sign_day=Month(Sign_date);
WEEKDAY(SAS-date)	Return the week day number (1-7) for a SAS date. 1 represents Sunday and so on.	Sign_WD=Weekday(Sign_date);

(d) An example on using MDY:

```
data abc; /*p4_mdy.sas*/
  Join_date=MDY(7,31,1997);
run;

proc print data=abc;
  format Join_date DDMYY;
run;
```

The output:

Obs	Join_date
1	31/07/97

(e) Commonly used numeric functions

Descriptive Statistics Functions	SAS action	Example
SUM(argument-1, argument-2, ..., argument-n)	Return the sum of the numeric arguments.	SumVar=SUM(Var1,Var2,Var3,Var4);
MEAN(argument-1, argument-2, ..., argument-n)	Return the average of the numeric arguments.	MeanVar=MEAN(OF Var1-Var4);
MIN(argument-1, argument-2, ..., argument-n)	Return the smallest value of the numeric arguments.	MinVar=MIN(Var1,Var2,Var3,Var4);

MAX(<i>argument-1</i> , <i>argument-2</i> , ..., <i>argument-n</i>)	*Return the largest value of the numeric <i>arguments</i> .	MaxVar=MAX(Var1,Var2,Var3,Var4);
N(<i>argument-1</i> , <i>argument-2</i> , ..., <i>argument-n</i>)	*Count of the non-missing <i>arguments</i> .	NVar=N(OF Var1-Var4);
NMISS(<i>argument-1</i> , <i>argument-2</i> , ..., <i>argument-n</i>)	*Count of the missing <i>NUMERIC arguments</i> .	NMissVar=NMISS(Var1,Var2,Var3,Var4);
CMISS(<i>argument-1</i> , <i>argument-2</i> , ..., <i>argument-n</i>)	*Count of the missing <i>NUMERIC or CHARACTER arguments</i> .	CMissVar=CMISS(Var1,Var2,Var3,Var4);

#Missing values are ignored.

*An argument can be a variable list, which is preceded by OF.




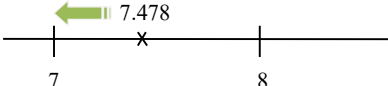
Ex. 4.7: Submit the following SAS program and fill in the blanks in the table.

```
* p4ex7.sas;
data Statistics;
  Var1=.;
  Var2=11;
  Var3=5;
  Var4=8;
  SumVar=SUM(Var1,Var2,Var3, Var4);
  MeanVar=MEAN(OF Var1-Var4);
  MinVar=MIN(Var1,Var2,Var3,Var4);
  MaxVar=MAX(Var1,Var2,Var3, Var4);
  NVar=N(OF Var1-Var4);
  NMissVar=NMISS(Var1,Var2,Var3,Var4);
  CMissVar=CMISS(Var1,Var2,Var3,Var4);
run;
```

Answers:

	Value
Total	
Average	
Minimum	
Maximum	
No. of non-missing	
No. of missing numeric values	
No. of missing numeric or character values	

Truncation Functions	SAS action	Example: Value1=7.478
ROUND(<i>argument</i> <, <i>round-off-unit</i> >)	Return a value rounded to the nearest multiple of the <i>round-off-unit</i> for the <i>argument</i> .	RDValue1=ROUND(Value1,0.1);
CEIL(<i>argument</i>)	Return the smallest integer greater than or equal to the <i>argument</i> .	CeilValue1=CEIL(Value1); 

<code>FLOOR(argument)</code>	Return the greatest integer less than or equal to the argument.	<code>FloorValue1=FLOOR(Value1);</code> 
<code>INT(argument)</code>	Return the integer part of argument.	<code>IntValue1=INT(Value1);</code>



Ex. 4.8: Submit the following SAS program and fill in the blanks in the table. When the variable `Value1=7.478`, what are the results using the truncation function? Do the functions, `INT` and `FLOOR` always give the same results?

```
data Truncation; /*p4ex8.sas*/
  Value1=7.478;
  RDValue1=ROUND(Value1,0.25);
  CeilValue1=CEIL(Value1);
  FloorValue1=FLOOR(Value1);
  IntValue1=INT(Value1);
run;
```


Answers:

	Truncated	Value1
Value1	7.478	-7.478
<code>ROUND(Value1,0.25)</code>		
<code>CEIL(Value1)</code>		
<code>FLOOR(Value1)</code>		
<code>INT(Value1)</code>		

(f) Commonly used character functions

Formatting Functions	SAS action	Example
<code>UPCASE(argument)</code>	Return the character in upper case of the argument.	<code>NSurname=UPCASE(Surname);</code>
<code>LOWCASE(argument)</code>	Return the character in lower case of the argument.	<code>LCase=LOWCASE(Name);</code>
<code>PROPCASE(argument <, delimiter(s)>)</code>	Convert all words in an argument to proper case. i.e. First letter is uppercase and the remaining letters are lowercase.	<code>Name='AX&BC MATHS';</code> <code>Pname1=propcase(Name);</code> <code>Pname2=propcase(Name, '&');</code> (<code>'Ax&bc Maths'</code> is assigned to <code>Pname1</code> and <code>'Ax&Bc Maths'</code> is assigned to <code>Pname2</code> .)
<code>RIGHT(string)</code>	Right align a character expression	<code>ProductCode=RIGHT(Code);</code>
<code>LEFT(string)</code>	Left align a character expression	<code>Name=LEFT(Name);</code>
<code>TRIM(string)</code>	Remove trailing blanks from <i>string</i> .	<code>A=TRIM(Name);</code>
<code>STRIP(string)</code>	Remove all leading and trailing blanks from <i>string</i> .	<code>B=STRIP(Name);</code>
<code>COMPBL(string)</code>	Remove multiple blanks from string by translating each occurrence of two or more consecutive blanks into a single blank.	<code>C=COMPBL(Name);</code>

COMPRESS(<i>source</i> <, <i>chars</i> >);	Remove the characters listed in <i>chars</i> argument from the source character string.	Phone='852-01234567'; CPhone=COMPRESS(Phone,'-' ' '); '85201234567' is assigned to CPhone.
--	---	---

 **Ex. 4.9:** Submit the following SAS program and fill in the blanks in the table. The variable is 'Phrase1'. See the changes in the phrase under different function operations.

```
Data char_data;                /*p4ex9.sas*/
  length phrase1 $40;
  phrase1=' sas programming  course ';
  upcase_ph=upcase(phrase1);
  propcase_ph=propcase(phrase1);
  lowercase_up_ph=lowercase(upcase(phrase1));
  Right_ph=right(phrase1);
  Left_ph=left(phrase1);
  trim_ph=trim(phrase1);
  Compbl_ph=compbl(phrase1);
  Compress_ph=compress(phrase1,'a');
run;


proc print data=char_data;
run;
```

Answers: Phrase1=' sas programming course ' (= 1 space)

Function	Content of function applied to Phrase 1 in data set*#
upcase	
lowercase(upcase(Phrase1))	
propcase	
Right	
Left	
Trim	
Compbl	
Compress	

*All functions applied to Phrase1 except Lowcase().

Open the data set directly in the SAS library for viewing the space.

 yellow highlighted blank is added as the length of the variable assigned to the variable is longer than the actual variable value.

Extracting /Replacing functions	SAS action	Example
CHAR(<i>string</i> , <i>position</i>)	Extract one character from a specified <i>position</i> in a character <i>string</i> .	ProvinceCode=CHAR(Area Code,1);
SUBSTR(<i>string</i> , <i>start</i> <, <i>length</i>)	Extract character(s) from the <i>string</i> , starting from the position no. , <i>start</i> . Number of character to be extracted is indicated by <i>length</i> . If <i>length</i> is omitted, the characters extracted are the remainder of the <i>string</i> .	Item_Type=substr(Code, 1,3) (If Code='978-1- 5994', '978' will be assigned to Item_Type)

<code>FIND(string, substring<, modifiers, startpos>);</code>	<p>Return the first position of the target <i>substring</i> in the <i>string</i>. 0 is returned when the <i>substring</i> is not found.</p> <ul style="list-style-type: none"> • <i>Modifier</i>: 'I' , 'T' or 'IT': I- case insensitive search T- ignore trailing blanks in the string and substring values. • <i>Startpos</i> indicates the starting position for searching. If it is omitted, <i>Startpos</i> has a default value of 1. A negative value represent a backward search. 	<pre>Text= 'Austria Germany US'; Pos1=FIND(Text,'us'); Pos2=FIND(Text,'US',I) ;</pre> <p>Both Pos1 and Pos2 equal 2.</p>
<code>LENGTH(argument)</code>	Return the length of a non-blank character string, excluding trailing blanks.	<pre>Clen=length(Code);</pre> <p>(If Code contains 'AB12 ' only, 4 will be assigned to Clen)</p>
<code>SCAN(string,n<, charlist>);</code>	<p>Return the <i>n</i>-th word of a character <i>string</i>.</p> <ul style="list-style-type: none"> • The word are delimited by <i>charlist</i>. Default delimiters in PC are blank, . (dot), +, , &, !, \$, *(asterisk), ;(semicolon), -(hyphen), /, %, ^. • A negative <i>n</i> selects the word from the end of the string. • A missing value is returned if there are fewer than <i>n</i> words in the string. • Length of the created variable is 200 bytes, unless it is defined in the LENGTH statement. 	<pre>Second=scan(Phrase,2,' ');</pre> <p>If Phrase contains 'Bread and jam ', 'and' is assigned to the variable Second.</p> <pre>X=Scan(Phrase,-1,' ') ⇒'jam'</pre> <pre>Y= Scan(Phrase,4) ⇒' ' (i.e. missing)</pre> <p>Second, X, Y are \$ 200! (length 200 bytes)</p>
<code>TRANWRD(source, target, replacement);</code>	Replace the <i>target</i> string in the <i>source</i> string with <i>replacement</i> .	<pre>Product=Transwrd(Produ ct,'Luci','Lucy');</pre> <p>If Product contains "Luci's plan", "Lucy's plan" is assigned to Product.</p>



Ex. 4.10: Fills in the blank.

```
data Scan_year;
  Text="Saint Dorothy's Day, February 6th, 2007";

  Year = ?
run;
```

Use double quote for string as apostrophe is used after year!

Ans:

Concatenation functions	SAS action	Example:
		FamilyName is `Luenberger` and FirstName is `David`. (=1 space)
CATX(separator, string-1,..., string-n)	Concatenate the <i>n</i> strings and insert the separator between strings. <ul style="list-style-type: none"> Note that leading and trailing blanks are removed from each argument. Length of the created variable is 200 bytes, unless it is defined in the LENGTH statement. 	FName=catx(' ', FirstName, FamilyName); `David Luenberger` is assigned to FName.
CAT(string-1,..., string-n) Or concatenation operator: !! or	Concatenate the <i>n</i> strings. No action is made on <i>n</i> strings before the concatenation.	FName=cat(' ', FirstName, FamilyName); FName= FirstName!! FamilyName `David Luenberger` is assigned to FName.
CATS(string-1,..., string-n)	Concatenate the <i>n</i> strings. The leading and trailing blanks are removed before concatenation.	FName=cats(' ', FirstName, FamilyName); `DavidLuenberger` is assigned to FName.
CATT(string-1,..., string-n)	Concatenate the <i>n</i> strings. The trailing blanks are removed before concatenation.	FName=catt(' ', FirstName, FamilyName); `David Luenberger` is assigned to FName.

Trailing blank
of 'David'
removed!



Ex. 4.11: A simple SAS program for testing concatenate functions:

```
data catfunctions; /* p4ex11.sas */
  FamilyName=' Luenberger ';
  FirstName=' David ';
  Catx_FName=Catx(' ', FirstName, FamilyName);
  Cat_FName=Cat(FirstName, FamilyName);
  Cats_FName=Cats(FirstName, FamilyName);
  Catt_FName=Catt(FirstName, FamilyName);
run;
```

- (a) Click the icon of the data set, CATFUNCTIONS in the explorer window and check the character values in the data set table.
- (b) Run the program for testing the above examples.

- SAS functions using variable list: A SAS variable list is an abbreviated method of referring to a group of variable names.
 - Numbered range

In PDV:

Quart1	Quart2	Var1	Quart3	Quart4
√	√		√	√

Total=Sum(of Quart1--Quart4)

(b) Name range

In PDV:

Quart1	Two	Q3	Fourth	Var5
√	√	√	√	

Total=Sum(of Quart1--Fourth)

(c) Name prefix

Tot1	Quart2	Tot2	Tot3	Tot4
√		√	√	√

Total=Sum(of Tot:)

(d) Special SAS name lists

In PDV:

Quart1	Quart2	Var1	Quart3	Quart4
N	N	\$	N	N
√	√		√	√

Total=Sum(of _Numeric_)

Ex. 4.12: A simple SAS program for testing variable list:



```
* p4ex12.sas;
data var_list;
  Dorol=1;
  Dorol2=2;
  Coro3='3';
  Aaron=3;
  Dorol4=4;
  Donald5=5;
  Total=sum(of Dorol-Dorol4);
  Sum_Doro=sum(of Doro:);
  Total_Num=sum( of _Numeric_);
run;

Proc print data=var_list;
run;
```

Fill in the blanks.

Variable	Value
Total	
Sum_Doro*	
Total_Num	

✚ Converting data type

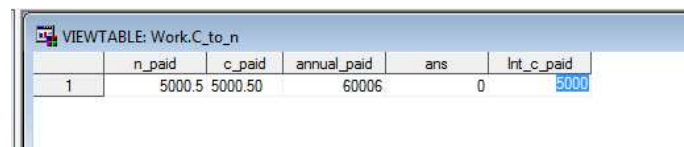
- From character to numeric
 - Automatic conversion
 - (a) happens when the character value is used in a numeric context, such as
 - (i) assignment to a numeric variable
 - (ii) an arithmetic operation
 - (iii) logical comparison with a numeric value. Note that WHERE statement and the WHERE= data set option do not perform any automatic conversion in comparisons.
 - (iv) a function that takes numeric arguments

The following SAS program show the above four cases and look at the data set. Note that numeric values are right-aligned while the character values are left-aligned.

‘C_paid’ is a character variable and ‘n_paid’ is declared as numeric variable in the LENGTH statement. ‘Annual_paid’ and ‘ans’ are not declared as any type in the program but become numeric after the compilation.

```
*p4example2.sas;
data c_to_n;
  length n_paid 8;
  c_paid='5000.50';
  * (i) ;
  n_paid=c_paid;
  * (ii) ;
  annual_paid = c_paid*12;
  * (iii) ;
  If c_paid< 5000 then ans=1;
  else ans=0;
  * (iv) ;
  Int_c_paid=int(c_paid);
run;
```

The output data set:



	n_paid	c_paid	annual_paid	ans	Int_c_paid
1	5000.5	5000.50	60006	0	5000

- (b) uses the w. informat.
- (c) produces a numeric missing value from a character value which does not conform to standard numeric notation(i.e. digits with optional decimal point and/or leading sign and/or the scientific notation with E in the expression)

- Use of INPUT function
 - (a) Syntax

```
NVar=INPUT(source, informat);
```

- (i) NVar is a numerical variable. Source refers to the source code to be converted into numeric. Note that the width in the informat is required. Refer to Chapter 2 for details of informat.
- (ii) No conversion message is written to the log.

(b) An example

```
*p4ex13.sas;
Data Input_con;
  Length ANVar1 8 ANVar2 8 ANVar3 8 ANVar4 8;
  CVar1='520000';
  CVar2='520.000';
  CVar3='11sep2001';
  CVar4='110901';
  NVar1=input(CVar1, 6.);
  NVar2=input(CVar2, comma7.);
  NVar3=input(CVar3, date9.);
  NVar4=input(CVar4, ddmmyy6.);
  ANVar1=CVar1;
  ANVar2=CVar2;
  ANVar3=CVar3;
  ANVar4=CVar4;
run;

proc print data=Input_con;
run;
```

(c) Comparison with automatic conversion



Ex. 4.13: Submit the above SAS program.

(a) Fill in the blanks in the table.

(b) Suggest a method to list the variable type and length of the above SAS program.

Answers: (a)

Character variable value	Conversion using INPUT	Automatic conversion
'520000'		
'520.000'*		
'11sep2001'		
'110901'		

*used in Germany for 520 thousand.

(b) Use PROC CONTENTS to list all variables and their attributes.

Output:



- Automatic conversion from another data set
- Converting a variable from numeric to character from a SAS data set to another data type
 - (a) Variable type cannot be changed within a DATA statement.

```
data assign; /*p4ex13.sas - no change change*/  
  CVar1='520000';  
  CVar1=input(CVar1,6.);  
run;
```

In the log file:

```
78  data assign;  
79  CVar1='520000';  
80  CVar1=input(CVar1,6.);  
81  run;  
  
NOTE: Numeric values have been converted to character values at the places given by:  
      (Line):(Column).  
      80:8  
NOTE: The data set WORK.ASSIGN has 1 observations and 1 variables.  
NOTE: DATA statement used (Total process time):  
      real time           0.01 seconds  
      cpu time            0.01 seconds
```

'CVar1' is defined as character type in the first assignment statement of CVar1. Although a note of the character to numeric conversion is written in the log file, there is **no change** in the variable type of CVar1 after the assignment statement.

(b) RENAME data set option cannot change the data type:

(i) Syntax of RENAME data set option

```
SAS-data-set (RENAME=(old-variable-name-1=new-variable-name-1
...
old-variable-name-n=new-variable-name-n));
```

(ii) One may consider the following steps for data type conversion:

1. Rename the target variable to a new name in the SET statement
2. Use INPUT function on the new variable for conversion of character to numeric value.

(iii) However, SAS cannot convert the data type of variable 'Payment' because it is defined as a character variable in the PDV, copied from the 'charpayment' variable type from abc.orginaldata. In this case, the numeric value produced from the INPUT function is converted into character, as 'Payment' is a character variable.

```
*conversion of the data type of "payment" from char to num;
data newdata;
  * (i) renaming the input variable, 'payment' as 'charpayment';
  set abc.orginaldata (rename=(payment=charpayment));
  * (ii) The INPUT function to convert the new variable into
  numeric type and assign the result into the old variable name;
  Payment=INPUT(Payment,comma6.);
run;
```

(c) Using DROP data set option for changing the data type:

(i) There are three steps for the conversion:

1. Read an input SAS data set.
2. Use INPUT function on the new variable for conversion of character to numeric variable.
3. Drop the old variable in the new data set in DATA statement.

(ii) An example: 'Payment' is a character variable in the data set, abc.orginaldata. Convert it into a numeric variable.

```
*(iii) Drop the old variable in the new data set;
data newdata(drop=charpayment);
  * (i) Read an SAS data set;
  set abc.orginaldata;
  * (ii) The INPUT function to convert the new variable into
  numeric type and assign the result into the old variable name;
  Payment=INPUT(Charpayment,comma6.);
run;
```

- From numeric to character

- Automatic conversion

(a) happens when the numeric value is used in a character context, such as

- (i) assignment to a character variable
- (ii) a concatenation operation, e.g CATX(...)
- (iii) a function which requires character arguments.
- (iv) Note that WHERE statement and the WHERE= data set option do not perform any automatic conversion in comparisons.

- (b) uses the BEST12. format. i.e. the length of the character variable is 12 and the resulting character value is right aligned.
- Use of PUT function
 - (a) Syntax

`CVar=PUT(source, format);`

- (i) *CVar* is a character variable. *Source* refers to the source code to be converted from numeric into characters. Note that the width in the format is required. Refer to Chapter 2 for details of SAS format (Section 2.6) and User-defined format (Section 2.7).
- (ii) No conversion message is written to the log.

(b) An example

```
data num_to_ch; /* p4ex14.sas */
length Ch_mobile $ 12;
  area_code=852;
  mobile=12345678;
  date_val=15229; * SAS numeric value of 11 Sep 2001;
  Ch_mobile =mobile;
  Phone0=('!!area_code!!')!!mobile;
  Phone1=cat('(',area_code,')',mobile);
  Phone2=cat('(',put(area_code,5.),')',mobile);
  Phone3=cat('(', '00', strip(area_code),')',mobile);
  * How to use '!!'? ;
  Date1=put(date_val,date9.);
  Date2=put(date_val,ddmmyy10.);
run;
```



Ex. 4.14: Submit the above SAS program. Fill in the blanks in the table.

Character variable	Content in data set
Ch_mobile	
Phone0	
Phone1	
Phone2	
Phone3	
Date1	
Date2	

- Similar to the case of changing variable type from character to numeric in a SAS data set, DROP data set option is used to convert a variable from numeric to character type of a SAS data set to another type. There are three steps for the conversion:
 - (i) Read an input SAS data set.
 - (ii) Use PUT function on the new variable for conversion of character to numeric variable.
 - (iii) Drop the old variable in the new data set in DATA statement.