SDAI
Q&A NOTES

July 2026

Q1)

1) Explain with an example how to match-merge two datasets in SAS where one dataset contains ID and name while another dataset contains ID and Score.
2) In a salary dataset, a programmer named the salary variable as "Sal". Write with example how to modify the "Sal" variable to "Salary"

ANSWER:

1) Using match-merge to combine two datasets

Data set **student**

ID	Name
1	Ravi
2	Meena

Data set **marks**

ID	Score
1	85
2	90

SAS Code

```
DATA final;
MERGE student marks;
BY ID;
RUN;
```

Output after merge: **final**

ID	Name	Score
1	Ravi	85
2	Meena	90

2) Renaming variable

Original data: **old**

EmpID	Sal
101	30000

SYNTAX is:

```
DATA new;
SET old;
RENAME oldname = newname;
RUN;
```

SAS Code

```
DATA employee_new;
SET employee;
RENAME Sal = Salary;
RUN;
```

Output data: **new**

EmpID	Salary
101	30000

Q2)

An employee_data dataset has two variables: EmpID and Salary. Assume it has two observations: (101, 30000) and (102, 40000). Write a SAS program to iteratively process the dataset to fill a new dataset "projection" with a new variable year and salary increasing by 1.1 times every year. Just show a sample output (values need not be accurate)

ANSWER:

Iteratively Processing Observations from a Data Set

Input data:

EmpID	Salary
101	30000
102	40000

For EmpID 101:

Year 1: 30000×1.10=33000

Year 2: 33000×1.10=36300

Year 3: 36300×1.10=39930

Output contains: 2×3=6 observations.

SAS Program:

```
3  DATA employee_data;
4      INPUT EmpID Salary;
5      DATALINES;
6  101 30000
7  102 40000
8  ;
9  RUN;
10
11 DATA projection;
12 SET employee_data;
13 DO year = 1 TO 3;
14    Salary = Salary * 1.10;
15    OUTPUT;
16 END;
17 RUN;
18
19 proc print data=projection;
20 run;
```

Sample Output:

Obs	EmpID	Salary	year
1	101	33000	1
2	101	36300	2
3	101	39930	3
4	102	44000	1
5	102	48400	2
6	102	53240	3

Q3)

Write SAS code to read DateOfBirth and LoginTime, from raw data and display them correctly. Write the output obtained.
Assume the following data:
Name, DOB, LoginTIme
Anu 15-AUG-2002 09:45:30
Bala 20-JAN-1998 14:20:15

ANSWER:

```
/* Reading Date and Time Values */

DATA Employee_Log;
    INPUT
        Name : $15.
        DateOfBirth : DATE9.
        LoginTime : TIME8.;

    FORMAT
        DateOfBirth DATE9.
        LoginTime TIME8.;
DATALINES;
Anu 15-AUG-2002 09:45:30
Bala 20-JAN-1998 14:20:15
;
RUN;

PROC PRINT DATA=Employee_Log;
    TITLE "Reading Date and Time Values";
RUN;
```

Sample Output:

Reading Date and Time Values

Obs	Name	DateOfBirth	LoginTime
1	Anu	15AUG2002	9:45:30
2	Bala	20JAN1998	14:20:15

Q4)

Explain the purpose of PROC FREQ in SAS. Create one-way frequency tables for Gender and Department, and interpret the frequency and percentage columns.

ANSWER:

PROC FREQ is a SAS procedure used to **count and summarize the frequency of categorical variables**.

It produces frequency tables that show how many observations belong to each category, along with percentages and cumulative statistics.

SAS Code for One-Way Frequency Tables

```
1  /*One-way-frequency table example */
2  data Employee;
3      input EmpID Gender $ Department $;
4      datalines;
5  101 M HR
6  102 F HR
7  103 M Sales
8  104 F Sales
9  105 M IT
10 106 F IT
11 107 M HR
12 108 F IT
13 109 M Sales
14 110 F HR
15 ;
16 run;
17
18 proc freq data=Employee;
19     tables Gender Department;
20     title "One-Way Frequency Tables";
21 run;
```

Sample Output:

One-Way Frequency Tables

The FREQ Procedure

Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	5	50.00	5	50.00
M	5	50.00	10	100.00

Department	Frequency	Percent	Cumulative Frequency	Cumulative Percent
HR	4	40.00	4	40.00
IT	3	30.00	7	70.00
Sales	3	30.00	10	100.00

Interpretation

Gender Frequency Table

- **Frequency** indicates the number of observations in each category.
  - **Male (M):** 5 employees
  - **Female (F):** 5 employees
- **Percent** shows the proportion of the total observations.
  - Males constitute **50%** of the employees.
  - Females constitute **50%** of the employees.
- This indicates that the dataset has an equal number of male and female employees.

Department Frequency Table

- **HR:** 4 employees (**40%**)
- **IT:** 3 employees (**30%**)
- **Sales:** 3 employees (**30%**)

From the frequency table:

- The **HR department has the highest number of employees**.
- **IT and Sales departments each contribute 30%** of the workforce.
- The cumulative percentage reaches **100%**, confirming that all observations have been accounted for.

Q5)

For the given data, write a SAS program using PROC REPORT to display the Sales - grouping by Region. Display the heading for Region as "Reg", Product as "Prod" and format Sales prefix

Region	Product	Sales
North	A	1200
North	B	1500
South	A	1800
South	B	2000

ANSWER:

SAS Program using PROC REPORT:

```
1  data sales;
2      input Region $ Product $ Sales;
3      datalines;
4  North A 1200
5  North B 1500
6  South A 1800
7  South B 2000
8  ;
9  run;
10
11 proc print data=sales;
12 run;
13
14 proc report data=sales ;
15     column Region Product Sales;
16
17     define Region / group "Reg";
18     define Product / display "Prod";
19     define Sales / display  format=dollar10.;
20
21 run;
```

Sample Output:

Obs	Region	Product	Sales
1	North	A	1200
2	North	B	1500
3	South	A	1800
4	South	B	2000

Reg	Prod	Sales
North	A	$1,200
	B	$1,500
South	A	$1,800
	B	$2,000

Q6)

1) Explain with an example what "Excluding unmatched observation" means.
2) How do you achieve merging two datasets like "inner join in SQL"? Explain with example.

ANSWER:

1) Excluding Unmatched Observations

Excluding unmatched observations means keeping only those observations that are present in all merged data sets.

```
DATA matched;
MERGE data1(IN=a) data2(IN=b);
BY ID;
IF a AND b;
RUN;
```

The IN= data set option creates temporary variables that indicate whether an observation comes from a particular data set.

Data set A:

ID	Name
1	Ravi
2	Meena
3	Arun

Data set B:

ID	Loan
1	50000
3	70000

Using:

```
DATA matched;
MERGE A(IN=a) B(IN=b);
BY ID;
IF a AND b;
RUN;
```

Output:

ID	Name	Loan
1	Ravi	50000
3	Arun	70000

ID 2 is excluded.

2) Merging similar to "Inner Join in SQL"

Selecting matching observations means selecting only records that have common BY variable values in two or more data sets.

SAS code:

```
DATA buyers;
MERGE registered(IN=r) purchase(IN=p);
BY ID;
IF r AND p;
RUN;
```

Explanation

This is similar to an inner join in SQL. Only observations that exist in both data sets are selected.

EXAMPLE:

Registered customers:

ID	Name
101	Asha
102	Bala
103	Charu

Purchase data:

ID	Amount
101	500
103	800
105	900

Matching IDs are 101 and 103.

Output data: buyers

ID	Name	Amount
101	Asha	500
103	Charu	800

Optional: START

```
MERGE registered(IN=r) purchase(IN=p);
```

- Combines two datasets:
  - registered
  - purchase
- SAS merges them **observation by observation** based on the BY variable.

What does IN= do?

IN= creates a temporary Boolean (0 or 1) variable.

registered(IN=r)

means

- r = 1 if the current observation comes from registered
- r = 0 otherwise

Similarly,

purchase(IN=p)

means

- p = 1 if the observation exists in purchase
- p = 0 otherwise

These variables exist only while the DATA step executes—they are **not** written to the output dataset.

```
IF r AND p;
```

This is the most important statement.
It means:
Keep the observation only if it exists in **both** datasets.

Optional: END

Q7)

A dataset old_marks contains the following values:

ID	Name	m1	m2	m3
101	Anu	65	70	75
102	Bob	55	60	65
103	Carol	85	90	95

Add 5 marks to each subject for all students and display the new dataset new_marks. Write the SAS program using array.

ANSWER:

Input:

ID	m1	m2	m3
101	65	70	75

Add 5 marks to each subject.

Output:

ID	m1	m2	m3
101	70	75	80

SAS Program:

```
1  DATA old_marks;
2      INPUT ID Name $ m1 m2 m3;
3      DATALINES;
4  101 Anu 65 70 75
5  102 Bob 55 60 65
6  103 Carol 85 90 95
7  ;
8  RUN;
9
10 DATA new_marks;
11 SET old_marks;
12 ARRAY score{3} m1 m2 m3;
13 DO i = 1 TO 3;
14    score{i} = score{i} + 5;
15 END;
16 RUN;
17
18 PROC PRINT DATA=new_marks;
19 RUN;
20
```

Sample Output:

Obs	ID	Name	m1	m2	m3	i
1	101	Anu	70	75	80	4
2	102	Bob	60	65	70	4
3	103	Carol	90	95	100	4

Q8)

1) Explain a SAS function with a general syntax.
2) List any 2 important SAS function rules.
3) List any 3 types of SAS functions and give one example each.

ANSWER:

1) Explain a SAS function with a general syntax.

A **SAS function** is a built-in routine that performs a specific operation on one or more arguments and returns a single value.

Basic Syntax

```
new_variable = function_name(argument);

new_variable = function_name(argument1, argument2, argument3);
```

2) List any 2 important SAS function rules.

1. SAS functions always return **one value**.
2. Function names are **not case-sensitive**.
3. Arguments can be **constants, variables, expressions, or other functions**.
4. Multiple functions can be **nested** inside one another.
5. Parentheses are **mandatory**, even if the function has a single argument.
6. Arguments are separated by **commas**.
7. Missing values may affect the result depending on the function

3) List any 3 types of SAS functions and give one example each.

Function Type	Examples
Numeric functions	`sum` , `mean` , `round`
Character functions	`upcase` , `substr` , `trim`
Date functions	`today` , `year` , `month`
Conversion functions	`input` , `put`
Mathematical functions	`sqrt` , `abs` , `log`
Missing value functions	`coalesce` , `nmiss` , `cmiss`

Q9)

Apply PROC MEANS to calculate descriptive statistics such as N, MEAN, MEDIAN, STD, MIN, and MAX for student marks. Take a sample size of 4 and write a SAS program.

ANSWER:

```
/* PROC MEANS for Descriptive Statistics */
data StudentMarks;
    input StudentID Name $ Marks;
    datalines;
101 Arun 80
102 Banu 70
103 Charan 60
104 Divya 90
;
run;

/* Calculate descriptive statistics */
proc means data=StudentMarks n mean median std min max;
    var Marks;
    title "Descriptive Statistics for Student Marks";
run;
```

Sample Output:

Descriptive Statistics for Student Marks

The MEANS Procedure

Analysis Variable : Marks					
N	Mean	Median	Std Dev	Minimum	Maximum
4	75.0000000	75.0000000	12.9099445	60.0000000	90.0000000

Q10)

For the given statistical data, construct executable SAS code using PROC SGPLOT to build a bar chart, a box plot, and a line plot.

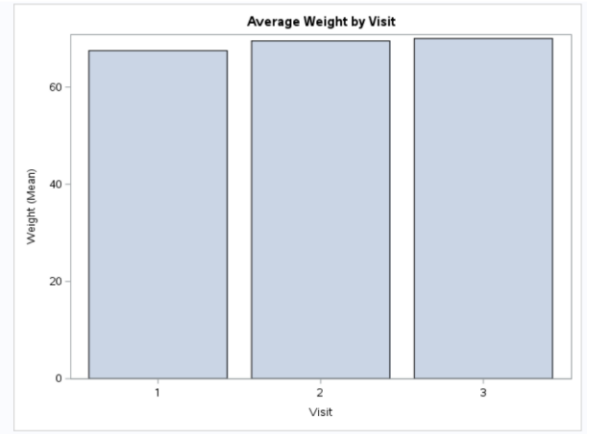
Visit	Patient	Weight
1	101	65
2	101	67
3	101	66
1	102	70
2	102	72
3	102	74

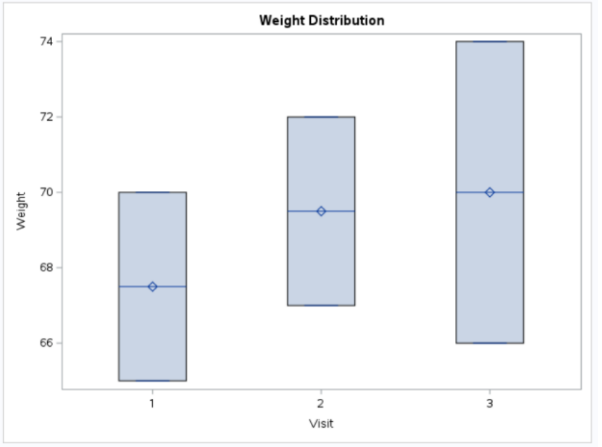
ANSWER:

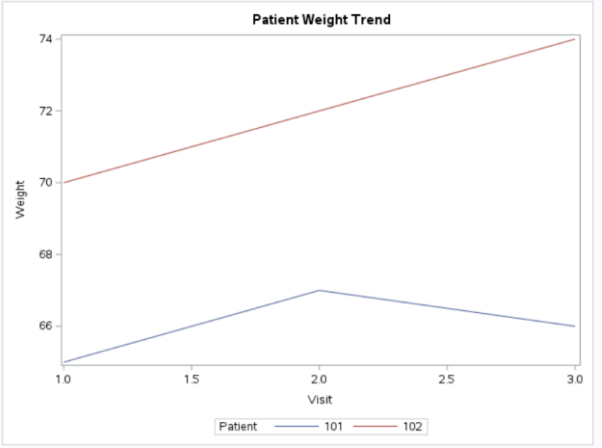
SAS Program using PROC SGPLOT:

```
*Build a bar chart, a box plot, and a line plot;
data clinic;
input Visit Patient Weight;
datalines;
1 101 65
2 101 67
3 101 66
1 102 70
2 102 72
3 102 74
;
run;

proc sgplot data=clinic;
vbar Visit / response=Weight stat=mean;
title "Average Weight by Visit";
run;

proc sgplot data=clinic;
vbox Weight / category=Visit;
title "Weight Distribution";
run;

proc sgplot data=clinic;
series x=Visit y=Weight / group=Patient;
title "Patient Weight Trend";
run;
```

Sample Output:







Q11)

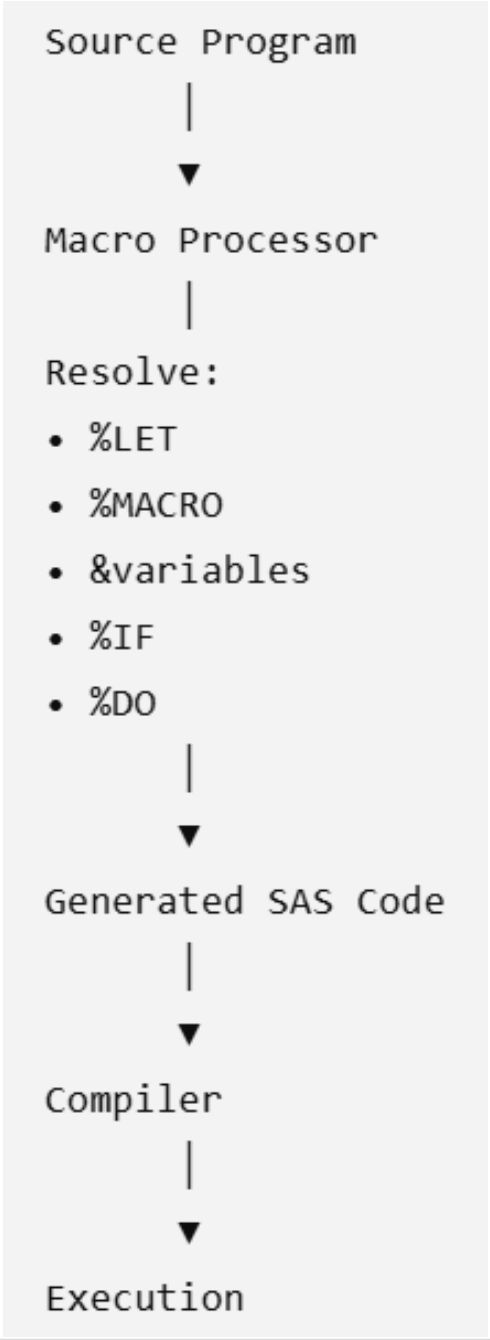
1) Classify spaghetti plots, survival plots, waterfall plots, and forest plots according to their primary analytical objectives and specialized applications in data analytics.
2) Explain SAS Macro Processing flow.

ANSWER:

1) Classify spaghetti plots, survival plots, waterfall plots, and forest plots according to their primary analytic\al objectives and specialized applications in data analytics.

Plot	Purpose	Application
Spaghetti Plot	Individual subject trends	Longitudinal clinical studies
Survival Plot	Time-to-event analysis	Clinical trials
Waterfall Plot	Individual response magnitude	Oncology trials
Forest Plot	Effect size comparison	Meta-analysis, subgroup analysis

2) Explain SAS Macro Processing flow.



Q12)

Distinguish between Global Macro Variable and Local Macro Variable Scopes in SAS. Give examples for both.

ANSWER:

Macro variables in SAS are stored in **symbol tables**. Their **scope** determines where they can be accessed and how long they remain available.

Feature	Global Macro Variable	Local Macro Variable
Definition	Created outside any macro using %LET or with CALL SYMPUTX (default global if no local table exists)	Created inside a macro using %LOCAL or as macro parameters
Scope	Available throughout the SAS session	Available only within the macro in which it is created
Lifetime	Exists until the SAS session ends or is deleted	Exists only while the macro executes
Accessibility	Can be referenced by all macros and DATA/PROC steps	Cannot be accessed outside the defining macro
Storage	Global symbol table	Local symbol table

Example 1: Global Macro Variable

```
%let Company = ABC Technologies;

%macro display;
   %put Company Name: &Company;
%mend;

%display;
```

Output

```
Company Name: ABC Technologies
```

Here, Company is stored in the **global symbol table** and is available throughout the session.

Example 2: Local Macro Variable

```
%macro student;
   %local Name;
   %let Name = Ravi;
   %put Student Name: &Name;
%mend;

%student;
%put Outside Macro: &Name;
```

Output

```
Student Name: Ravi
WARNING: Apparent symbolic reference NAME not resolved.
Outside Macro: &Name
```

The variable **Name** exists only while the macro executes.

Q13)

A university wants to automate student result processing using SAS. Explain how a) DO loops, b) DO WHILE and c) DO UNTIL can be used to process and present student data efficiently. Illustrate each concept with suitable SAS programs.

ANSWER:
SAS provides looping constructs, arrays, formats, and informats to automate repetitive processing and improve data presentation.

a. Basic DO Loop

A DO loop repeats statements for a specified number of iterations.

```
data Squares;
do Number=1 to 5;
Square=Number**2;
output;
end;
run;
```

Sample Output:

Number	Square
1	1
2	4
3	9
4	16
5	25

b. DO WHILE Statement

Checks the condition before execution.

```
data CountWhile;
x = 1;

do while (x <= 5);
   Square = x*x;
   output;
   x + 1;
end;
run;

proc print data=CountWhile;
run;
```

Sample Output:

Obs	x	Square
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

Explanation:

- The condition `x <= 5` is checked **before** each iteration.
- The loop stops when `x` becomes **6**.
- Therefore, **5 observations** are produced.

c. DO UNTIL Statement

Checks the condition after execution.

```
data CountUntil;
x = 1;

do until (x > 5);
   Square = x*x;
   output;
   x + 1;
end;
run;

proc print data=CountUntil;
run;
```

Sample Output:

Obs	x	Square
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

Explanation:

•The statements inside the loop execute **first**, and then the condition `x > 5` is checked.

•Since the condition is evaluated **after** the loop body, a **DO UNTIL** loop always executes **at least once**, even if the condition is initially true.

Summary：

Feature	DO WHILE	DO UNTIL
Condition checked	Before loop execution	After loop execution
Minimum iterations	0	1
Executes when condition	Is **true**	Is **false** (continues until it becomes true)
Common use	When the loop may not need to execute	When the loop should execute at least once

Q14)

a) Explain Output Delivery System (ODS) in SAS with syntax?
b) Write sample SAS code to export PROC PRINT Output to HTML
c) Write sample SAS code to export to Excel

ANSWER:

a) Explain Output Delivery System (ODS) in SAS with syntax?

The **Output Delivery System (ODS)** in SAS is a feature that allows users to generate procedure outputs in different formats such as **HTML, PDF, RTF, Excel, CSV, and others**.

ODS separates the output from its presentation, enabling reports to be shared easily across different platforms.

SYNTAX:

```
ODS destination FILE="filename.extension";

PROC procedure DATA=dataset;
   statements;
RUN;

ODS destination CLOSE;
```

Where:

- **destination** – Output format (HTML, PDF, RTF, EXCEL, etc.)
- **FILE=** – Specifies the output file name and location.
- **ODS CLOSE** – Closes the destination and completes the file.

b) Write sample SAS code to export PROC PRINT Output to HTML

```
ods html file="C:\Output\Report.html";

proc print data=students;
   title "Student Marks Report";
run;

ods html close;
```

Output
A web page (**Report.html**) is created containing the PROC PRINT report.

c) Write sample SAS code to export to Excel

```
ods excel file="C:\Output\Report.xlsx";

proc print data=students;
   title "Student Marks";
run;

ods excel close;
```

Output
An Excel workbook (**Report.xlsx**) containing the report.

Q15)

A retail company maintains separate SAS data sets for customer details and purchase records. Demonstrate using suitable SAS programs:
a) One-to-One Reading b) Concatenation, and c) Match-Merging

ANSWER:

(a) One-to-One Reading
Combines observations based on observation numbers.

```
data Combined;
set Customer;
set Purchase;
run;
```

Characteristics
•No common variable required.
•Suitable only when observations correspond by position.

Customer:

Customer_ID	Name
101	Ravi
102	Priya
103	Kumar

Purchase:

Amount	City
5000	Chennai
6500	Coimbatore
7200	Madurai

```
data Combined;
set Customer;
set Purchase;
run;
```

Sample Output (**Combined**):

Customer_ID	Name	Amount	City
101	Ravi	5000	Chennai
102	Priya	6500	Coimbatore
103	Kumar	7200	Madurai

Explanation:
The first observation from **Customer** is combined with the first observation from **Purchase**, the second with the second, and so on. No common variable is required.

(b) Concatenation

Stacks one data set below another.

```
data Sales_All;
set Sales_Jan Sales_Feb;
run;
```

Sales_Jan:

Month	Sales
January	5000
January	7000

Sales_Feb:

Month	Sales
February	6000
February	8000

Sample Output:

Sales_All:

Month	Sales
January	5000
January	7000
February	6000
February	8000

(c) Match-Merging

Requires sorting by the common key.

```
proc sort data=Customer;
by Customer_ID;
run;

proc sort data=Purchase;
by Customer_ID;
run;

data CustomerSales;
merge Customer Purchase;
by Customer_ID;
run;
```

Customer:

Customer_ID	Name
101	Ravi
102	Priya

Purchase:

Customer_ID	Amount
101	5000
102	6500

CustomerSales:

Customer_ID	Name	Amount
101	Ravi	5000
102	Priya	6500

Summary :

Method	Purpose	Requires Common Variable
One-to-One Reading	Combine by observation number	No
Concatenation	Append observations	No
Match-Merging	Combine related records	Yes

Q16)

a) Distinguish INPUT and PUT functions for data conversion in SAS.
b) Write a SAS program to explain INPUT function.
c) Write a SAS program to explain PUT function.

ANSWER:

a) Distinguish INPUT and PUT functions for data conversion in SAS.

Feature	INPUT Function	PUT Function
Purpose	Converts character to numeric	Converts numeric to character
Uses	Informat	Format
Result	Numeric value	Character value
Syntax	input(char_var, informat.)	put(num_var, format.)
Example	Age=input(AgeChar,3.);	AgeChar=put(Age,3.);

b) Write a SAS code to explain PUT function.

```
1  /* Convert Numeric to Character using PUT */
2  data Employee;
3      input Name $ Salary;
4      SalaryChar = put(Salary, 8.);
5      datalines;
6  Anu 50000
7  Bala 40000
8  ;
9  run;
10
11 proc print data=Employee;
12 run;
```

Sample Output:

Obs	Name	Salary	SalaryChar
1	Anu	50000	50000
2	Bala	40000	40000

c) Write a SAS code to explain INPUT function.

```
1  /* Convert Character to Numeric using INPUT */
2  data Employee;
3      input Name $ AgeChar $;
4      Age = input(AgeChar, 3.);
5      datalines;
6  John 25
7  Mary 30
8  David 28
9  ;
10 run;
11
12 proc print data=Employee;
13 run;
```

Sample Output:

Obs	Name	AgeChar	Age
1	John	25	25
2	Mary	30	30
3	David	28	28