LECTURE 06

21CSE295T

Statistical Data Analytics And Integration

Unit 3 Advanced Statistical Techniques in SAS

3.1O	Basics of DO Loops
3.11	Executing DO Loops
3.12	OUTPUT Statements
3.13	Nesting DO Loops
3.14	Iteratively Processing Observations from a Data Set
3.15	DO UNTIL Statement
3.16	DO WHILE Statement
3.17	Defining and Processing Arrays
3.18	Applying SAS Formats and Informats
3.19	Specifying SAS Formats

Virupakshan K

+91-9840847872 | VIRU@KART.LA

SYLLABUS
21CSE295T Statistical Data Analytics And Integration

Unit 1 Introduction to Statistical Analysis — **9 hours**

Computational Statistics- Statistical Pattern Recognition, Nonparametric Regression, Exploratory Data Analysis, Monte Carlo Methods for Inferential Statistics-Basic concepts of statistics, Types of data (categorical, numerical), Populations and samples, Measures of central tendency (mean, median, mode), Measures of variability (range, variance, standard deviation), Descriptive vs. inferential statistics.

Tutorial:

- SAS Statements, Structure, Libraries, Rules, Datasets, Representation of missing values, Assigning Libraries, PROC CONTENTS, visualize the descriptor and datasets attributes

Unit 2 Data Analysis Using SAS — **9 hours**

Basic Reports – Selecting Variables – Identifying observations – Subsetting the Data – Limiting the observations-SORT Procedure – Generating Column totals – Specifying titles and footnotes in procedure output – Assigning Descriptive labels and permanent label -How SAS Processes Programs – Compilation Phase – PDV – Descriptor Portion-Execution Phase – Data Portion – Debugging a Data Step – Testing Programs-BY-Group Processing Definitions – Preprocessing Data – Determine Whether the Data Requires Preprocessing – Sorting Observations for BY-Group Processing-FIRST. and LAST. DATA Step Variables – How the DATA Step Identifies BY Groups – Assignment Statements – Operators in SAS Expressions

Tutorial:

- Introduction of report and PRINT Procedure-Hands on practice of VAR, ID, WHERE statement, where expression[eq, ne, gt, lt, ge, le, AND, OR, CONTAINS]

Unit 3 Advanced Statistical Techniques in SAS — **9 hours**

Determining the Structure and Contents of Data Sets – Testing Program – Methods of Combining SAS Data Sets-One-to-One Reading – Concatenating – Match-Merging – Renaming Variables – Excluding Unmatched Observations -Selecting Matching Observations -Basics of DO Loops – Executing DO Loops -OUTPUT Statements – Nesting DO Loops-Iteratively Processing Observations from a Data Set -DO UNTIL Statement – DO WHILE Statement Defining and Processing Arrays – Arrays -Applying SAS Formats and Informats – Specifying SAS Formats

Tutorial:

- One-to-One Reading to Combine Data Sets-Concatenating to Combine Data Sets-Match-Merging to Combine Data Sets-Renaming Variables-Processing Iterative DO Loops-Indenting and Nesting DO Groups

Unit 4 SAS Programming and Automation — **9 hours**

Definitions – Reading Date and Time with Informats – Displaying Date and Time Values with Formats-Basics of SAS Functions - SAS Functions Syntax – Converting Data with Functions- The MEANS Procedure – Specifying Descriptive Statistics – Limiting Decimal Places – Specifying Variable Using VAR Statement – Summarized Data Set Using OUTPUT Statement – The FREQ Procedure-Specifying Variable Using TABLES Statements - Two-Way and N-Way Tables – LIST Options-The Output Delivery System (ODS) – HTML Outputs – PDF Outputs-RTF Outputs – EXCEL Outputs – The EXPORT Procedure

Tutorial:

- Reading Date and Time with Formats and Informats- Execution of Data Values Using Numeric Functions-Creating a Descriptive Statistics by Using PROC MEANS-Creating a Descriptive Statistics by Using PROC FREQ-Creating Outputs with HTML, PDF, RTF, EXCEL Formats-Exporting a External File

Unit 5 Advanced SAS Topics — **9 hours**

Creating a Default List Report – Windowing/Non Windowing Mode - Define Statement-Defining Variables – Column Statement – Selecting Observations – Defining the Variables-Group Processing the Variables by Using Different Options – Computing the Variable-Bar Chart – Box Plot – Mean Measurement Over the Time using Line Plot – Spaghetti Plot – Survival Plot – Waterfall Plot – Forest Plot-Introduction to Macro Facility – SAS Programs and Macro Processing – Macro Variables – Macro Processing – Scopes of Macro Variables – Macro Expressions-Scopes of Macro Variables – Macro Expressions-Macro Quoting – Interfaces with Macro Facility -Storing and Reusing Macros.

Tutorial:

- Creating Graphs by Using Bar Chart -Creating Graphs Using Various Types of Plots-Generating SAS Code Using Macros-Defining User-Defined Macro Variables-Creating a Tables Using PROC SQL with different Clauses- Generating a Cartesian Product -Combining Tables Using Multiple Joins

LECTURE 06 TOPICS

Unit 3 Advanced Statistical Techniques in SAS

3.1	Determining the Structure and Contents of Data Sets
3.2	Testing Program
3.3	Methods for Combining SAS Data Sets
3.4	One-to-One Reading
3.5	Concatenating
3.6	Match-Merging
3.7	Renaming Variables
3.8	Excluding Unmatched Observations
3.9	Selecting Matching Observations
3.1O	Basics of DO Loops
3.11	Executing DO Loops
3.12	OUTPUT Statements
3.13	Nesting DO Loops
3.14	Iteratively Processing Observations from a Data Set
3.15	DO UNTIL Statement
3.16	DO WHILE Statement
3.17	Defining and Processing Arrays
3.18	Applying SAS Formats and Informats
3.19	Specifying SAS Formats

3.1O	Basics of DO Loops
3.11	Executing DO Loops
3.12	OUTPUT Statements
3.13	Nesting DO Loops
3.14	Iteratively Processing Observations from a Data Set
3.15	DO UNTIL Statement
3.16	DO WHILE Statement
3.17	Defining and Processing Arrays
3.18	Applying SAS Formats and Informats
3.19	Specifying SAS Formats

3.10.1 Basics of DO Loops

A DO loop in SAS is used to execute a group of statements repeatedly.

```
DO index-variable = start TO stop BY increment;
   statements;
END;
```

A DO loop reduces repeated coding. It is mainly used for iterative calculations.

Number of iterations:

$$\text{Iterations} = \frac{\text{Stop} - \text{Start}}{\text{Increment}} + 1$$

CODE | LOG | RESULTS | OUTPUT DATA

```
1  * Do example by Viru;
2  DATA numbers;
3     DO i = 1 TO 5;
4        square = i*i;
5        OUTPUT;
6     END;
7  RUN;
8
9  PROC PRINT DATA=numbers;
10 RUN;
```

3.10.2 Basics of DO Loops: Example 1

CODE LOG RESULTS OUTPUT DATA

```
1  * Do example by Viru;
2  DATA numbers;
3     DO i = 1 TO 5;
4        square = i*i;
5        OUTPUT;
6     END;
7  RUN;
8
9  PROC PRINT DATA=numbers;
10 RUN;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	i	square
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

CODE LOG RESULTS OUTPUT DATA

Table: WORK.NUMBERS | View: Column names

Filter: (none)

Columns: Select all, i, square

Total rows: 5 Total colu… Rows 1-5

	i	square
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

3.10.3 Basics of DO Loops: Example 2 (comment OUTPUT;)

CODE LOG RESULTS OUTPUT DATA

```
1  * Do example 2 by Viru;
2  DATA numbers;
3     DO i = 1 TO 5;
4        square = i*i;
5        *OUTPUT;
6     END;
7  RUN;
8
9  PROC PRINT DATA=numbers;
10 RUN;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	i	square
1	6	25

CODE LOG RESULTS OUTPUT DATA

Table: WORK.NUMBERS | View: Column names

Filter: (none)

Columns

- Select all
- i
- square

Total rows: 1 Tot... Rows 1-1

	i	s
1	6	

3.11.1 Executing DO Loops

Definition

Executing `DO` loops means repeatedly running statements inside the `DO-END` block according to specified conditions.

Syntax

```
DO i = 1 TO 10;
    statements;
END;
```

Explanation

The loop begins with the initial value of the index variable and continues until the ending value is reached.

3.11.2a Executing DO Loops: Example 1

Real-World Example

A finance company calculates yearly investment value for 10 years.

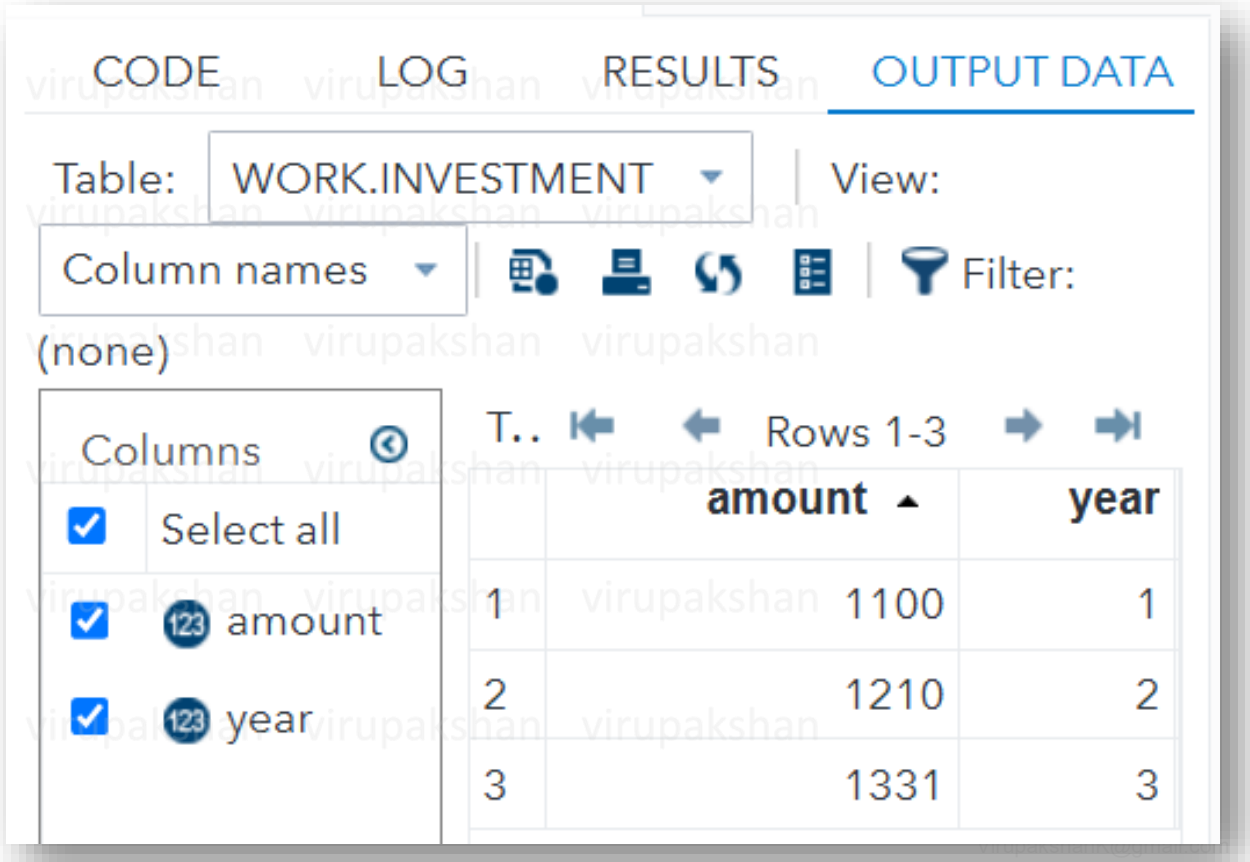
Solved Numerical Example

Initial amount = 1000

Interest rate = 10%

Number of years = 3

```
DATA investment;
amount = 1000;
DO year = 1 TO 3;
    amount = amount * 1.10;
    OUTPUT;
END;
RUN;
```



	amount	year
1	1100	1
2	1210	2
3	1331	3

3.11.2b Executing DO Loops: Example 1

CODE LOG RESULTS OUTPUT DATA

```
1  * Do statement example 1 by Viru;
2  DATA investment;
3  amount = 1000;
4  DO year = 1 TO 3;
5     amount = amount * 1.10;
6     OUTPUT;
7  END;
8  RUN;
9
10 proc print data=investment;
11 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	amount	year
1	1100	1
2	1210	2
3	1331	3

CODE LOG RESULTS OUTPUT DATA

Table: WORK.INVESTMENT | View:

Column names | Filter:

(none)

Columns

Select all

amount

year

T.. Rows 1-3

	amount	year
1	1100	1
2	1210	2
3	1331	3

3.11.2c Executing DO Loops: Example 1 – comment OUTPUT

CODE LOG RESULTS OUTPUT DATA

```
1  * Do statement example 2 by Viru;
2  DATA investment;
3  amount = 1000;
4  DO year = 1 TO 3;
5     amount = amount * 1.10;
6     *OUTPUT;
7  END;
8  RUN;
9
10 proc print data=investment;
11 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	amount	year
1	1331	4

CODE LOG RESULTS OUTPUT DATA

Table: WORK.INVESTMENT | View:

Column names | Filter:

(none)

Columns

- Select all
- amount
- year

Rows 1-1

	amount	year
1	1331	4

3.12.1 OUTPUT Statements

Definition

The `OUTPUT` statement explicitly writes the current observation to a SAS data set.

Syntax

```
OUTPUT;
```

Explanation

By default, SAS outputs one observation at the end of each DATA step iteration.

But inside loops, `OUTPUT` is needed to create multiple observations.

```
DATA sample;
DO i = 1 TO 3;
    value = i * 10;
    OUTPUT;
END;
RUN;
```

Output:

i	value
1	10
2	20
3	30

Without `OUTPUT`, only the final value may be written.

3.12.2 OUTPUT Statements: Example 1

```
1  * Output statement example 1 by Viru;
2  DATA myDataSet;
3  sum =0;
4  DO i = 1 TO 3;
5     sum + i;
6     OUTPUT;
7  END;
8  RUN;
9
10 proc print data=myDataSet;
11 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	sum	i
1	1	1
2	3	2
3	6	3

CODE LOG RESULTS OUTPUT DATA

Table: WORK.MYDATASET | View:

Column names | Filter: (none)

Columns: Select all, sum, i

T... Rows 1-3

	sum	i
1	1	1
2	3	2
3	6	3

3.12.3 OUTPUT Statements: Example 2 (without OUTPUT)

CODE LOG RESULTS OUTPUT DATA

```
1  * Output statement example 2 by Viru;
2  DATA myDataSet;
3  sum =0;
4  DO i = 1 TO 3;
5     sum + i;
6     *OUTPUT;
7  END;
8  RUN;
9
10 proc print data=myDataSet;
11 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	sum	i
1	6	4

CODE LOG RESULTS OUTPUT DATA

Table: WORK.MYDATASET | View:

Column names | Filter:

(none)

Columns

- Select all
- sum
- i

To... Rows 1-1

	sum	i
1	6	4

3.13.1 Nesting DO Loops

Definition

Nesting DO loops means placing one DO loop inside another.

Syntax

```
DO i = 1 TO 3;
    DO j = 1 TO 2;
        statements;
        OUTPUT;
    END;
END;
```

Explanation

Nested loops are useful when multiple levels of repetition are required.

Formula

Total iterations

=Outer loop iterations × Inner loop iterations

3.13.2 Nesting DO Loops

```
DATA nested;
DO region = 1 TO 2;
    DO month = 1 TO 3;
        sales = region * month * 100;
        OUTPUT;
    END;
END;
RUN;
```

Output:

Region	Month	Sales
1	1	100
1	2	200
1	3	300
2	1	200
2	2	400
2	3	600

Total observations:
2×3=6

3.13.3 Nesting DO Loops: Example 1

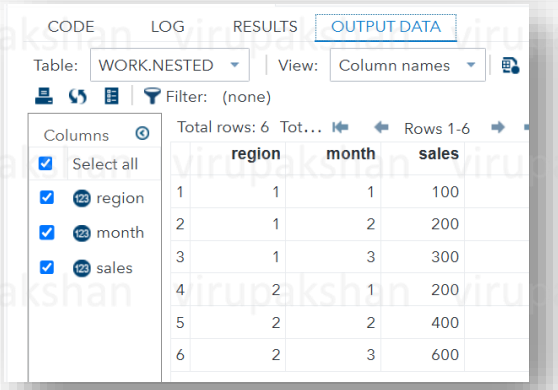
CODE LOG RESULTS OUTPUT DATA

```
1  * Example for Nested DO loop by Viru;
2  DATA nested;
3  DO region = 1 TO 2;
4     DO month = 1 TO 3;
5        sales = region * month * 100;
6        OUTPUT;
7     END;
8  END;
9  RUN;
10
11 proc print data=nested;
12 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	region	month	sales
1	1	1	100
2	1	2	200
3	1	3	300
4	2	1	200
5	2	2	400
6	2	3	600



3.14.1 Iteratively Processing Observations from a Data Set

Definition

Iteratively processing observations means using loops to process multiple records or repeated values from a data set.

Syntax

```
DATA new;
SET old;
DO i = 1 TO n;
   statements;
END;
RUN;
```

Explanation

Each observation from an input data set can be processed repeatedly using a loop.

3.14.2 Iteratively Processing Observations from a Data Set

Solved Numerical Example

Input data:

EmpID	Salary
101	30000
102	40000

```
DATA projection;
SET employees;
DO year = 1 TO 3;
   Salary = Salary * 1.10;
   OUTPUT;
END;
RUN;
```

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.

3.14.3a Iteratively Processing Observations from a Data Set

CODE LOG RESULTS OUTPUT DATA

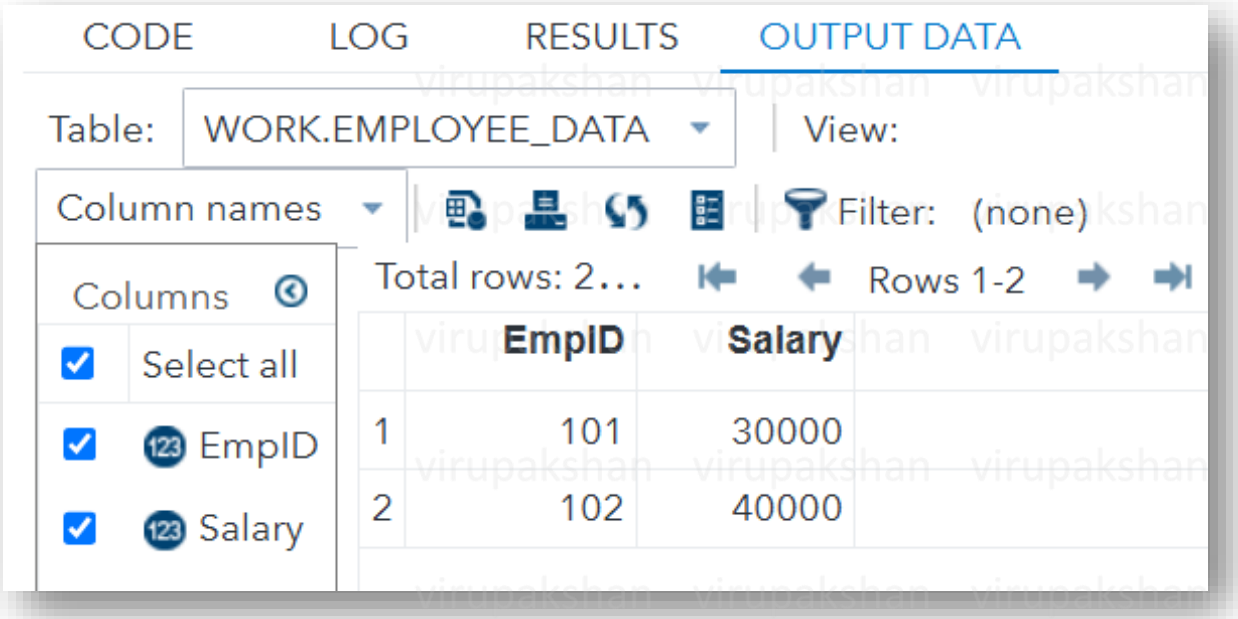
```
1  * DO Loop example 2 by Viru;
2
3  DATA employee_data;
4      INPUT EmpID Salary;
5      DATALINES;
6  101 30000
7  102 40000
8  ;
9  RUN;
10
11 DATA projection;
12 SET employee_data;
13 DO year = 1 TO 3;
14    Salary = Salary * 1.10;
15    OUTPUT;
16 END;
17 RUN;
18
19 proc print data=projection;
20 run;
```

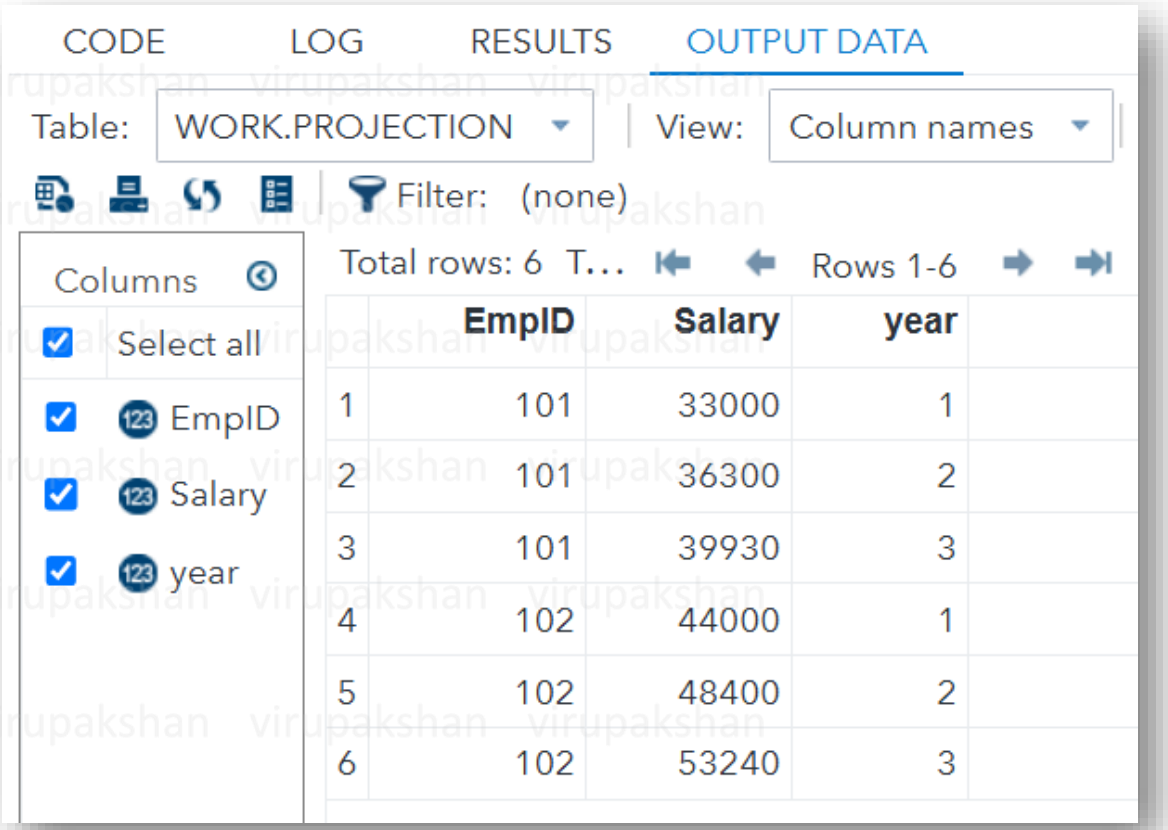
CODE LOG RESULTS OUTPUT DATA

Table of Contents

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

3.14.3b Iteratively Processing Observations from a Data Set





3.15.1 DO UNTIL Statement

Definition

The `DO UNTIL` loop executes statements repeatedly until a condition becomes true. It always executes at least once.

Syntax

```
DO UNTIL(condition);
    statements;
END;
```

Explanation

The condition is checked at the end of the loop.

Key Point

`DO UNTIL` executes at least one time even if the condition is true initially.

3.15.2 DO UNTIL Statement

Solved Numerical Example

Initial amount = 5000

Deposit each time = 1000

Stop when amount reaches at least 8000

```
DATA savings;
amount = 5000;
DO UNTIL(amount >= 8000);
    amount = amount + 1000;
    OUTPUT;
END;
RUN;
```

Iterations:

1.5000 + 1000 = 6000

2.6000 + 1000 = 7000

3.7000 + 1000 = 8000

Output has 3 observations.

3.15.3 DO UNTIL Statement

CODE LOG RESULTS OUTPUT DATA

Line #

```
1  * DO UNTIL example by Viru;
2
3  DATA savings;
4  amount = 5000;
5  DO UNTIL(amount >= 8000);
6     amount = amount + 1000;
7     OUTPUT;
8  END;
9  RUN;
10
11 PROC PRINT Data=savings;
12 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	amount
1	6000
2	7000
3	8000

CODE LOG RESULTS OUTPUT DATA

Table: WORK.SAVINGS | View: Column names

Filter: (none)

Columns

Select all

amount

Tot... Rows 1-3

	amount
1	6000
2	7000
3	8000

3.16.1 DO WHILE Statement

Definition

The `DO WHILE` loop executes statements while a condition is true. The condition is checked before loop execution.

Syntax

```
DO WHILE(condition);
    statements;
END;
```

Explanation

If the condition is false initially, the loop will not execute even once.

3.16.2 DO WHILE Statement

Solved Numerical Example

Initial amount = 5000

Deposit = 1000

Continue while amount < 8000

```
DATA savings;
amount = 5000;
DO WHILE(amount < 8000);
    amount = amount + 1000;
    OUTPUT;
END;
RUN;
```

Output:

Amount
6000
7000
8000

3.16.3 DO WHILE Statement

CODE LOG RESULTS OUTPUT DATA

Line #

```
1  * DO WHILE example by Viru;
2
3  DATA savings;
4  amount = 5000;
5  DO WHILE(amount < 8000);
6      amount = amount + 1000;
7      OUTPUT;
8  END;
9  RUN;
10
11
12 PROC PRINT Data=savings;
13 run;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	amount
1	6000
2	7000
3	8000

CODE LOG RESULTS OUTPUT DATA

Table: WORK.SAVINGS | View: Column names

Filter: (none)

Columns

Select all

amount

Rows 1-3

	amount
1	6000
2	7000
3	8000

3.16.4 Difference Between DO WHILE and DO UNTIL

Feature	DO WHILE	DO UNTIL
Condition checked	Before execution	After execution
Minimum execution	Zero times	At least once
Runs when condition is	True	False until condition becomes true

```
DO WHILE(condition);
    statements;
END;
```

```
DO UNTIL(condition);
    statements;
END;
```

3.17.1 Defining and Processing Arrays

Definition

An array in SAS is a temporary grouping of variables that allows similar variables to be processed together.

Syntax

```
ARRAY array-name {n} variable-list;
```

Example:

```
ARRAY marks{3} mark1 mark2 mark3;
```

```
Array: marks

marks{1}  ---->  mark1
marks{2}  ---->  mark2
marks{3}  ---->  mark3
```

- **ARRAY** – SAS keyword used to define an array.
- **marks** – Name of the array (not a dataset variable).
- **{3}** – Number of elements in the array.
- **mark1 mark2 mark3** – Variables that are members of the array.

The array exists only while the DATA step executes; it is **not stored** in the output dataset.

3.17.2 Defining and Processing Arrays

Solved Numerical Example

Input:

ID	m1	m2	m3
1	70	80	90

Add 5 marks to each subject.

```
DATA updated;
SET marks;
ARRAY score{3} m1 m2 m3;
DO i = 1 TO 3;
   score{i} = score{i} + 5;
END;
RUN;
```

Output:

ID	m1	m2	m3
1	75	85	95

3.17.3a Defining and Processing Arrays

CODE LOG RESULTS OUTPU

```
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=old_marks;
19 RUN;
20
21 PROC PRINT DATA=new_marks;
22 RUN;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	ID	Name	m1	m2	m3
1	101	Anu	65	70	75
2	102	Bob	55	60	65
3	103	Carol	85	90	95

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

3.17.3b Defining and Processing Arrays

CODE LOG RESULTS OUTPUT DATA

Table: WORK.OLD_MARKS | View: Column names

Filter: (none)

Columns

- Select all
- ID
- Name
- m1
- m2
- m3

Total rows: 3 Tota... Rows 1-3

	ID	Name	m1	m2	m3
1	101	Anu	65	70	75
2	102	Bob	55	60	65
3	103	Carol	85	90	95

CODE LOG RESULTS OUTPUT DATA

Table: WORK.NEW_MARKS | View: Column names

Filter: (none)

Columns

- Select all
- ID
- Name
- m1
- m2
- m3
- i

Total rows: 3 Tota... Rows 1-3

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

3.18.1 Applying SAS Formats and Informats

Definition

Formats control how data values are displayed.

Informats tell SAS how to read raw data values.

Difference

Aspect	Format	Informat
Purpose	Displays data	Reads data
Used while	Output/display	Input/reading
Example	DATE9.	MMDDYY10.

Syntax

```
FORMAT variable format.;
INFORMAT variable informat.;
```

Explanation

- Format changes appearance, not stored value.
- Informat helps SAS correctly interpret raw data.

3.18.2 Applying SAS Formats and Informats

Solved Numerical Example

```
DATA date_example;
INPUT name $ dob MMDDYY10.;
FORMAT dob DATE9.;
DATALINES;
Ravi 07/15/2000
Meena 12/25/1999
;
RUN;
```

SAS reads `07/15/2000` using `MMDDYY10.` and displays it as `15JUL2000`.

3.18.3 Applying SAS Formats and Informats

CODE LOG RESULTS OUTPUT DATA

```
1  * FORMAT example by Viru;
2  DATA date_example;
3  INPUT name $ dob MMDDYY10.;
4  FORMAT dob DATE9.;
5  DATALINES;
6  Ravi 07/15/2000
7  Meena 12/25/1999
8  ;
9  RUN;
10
11
12 PROC PRINT DATA=date_example;
13 RUN;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	name	dob
1	Ravi	15JUL2000
2	Meena	25DEC1999

CODE LOG RESULTS OUTPUT DATA

Table: WORK.DATE_EXAMPLE | View: Column names

Filter: (none)

Columns: Select all, name, dob

Total rows: 2 Tota... Rows 1-2

	name	dob
1	Ravi	15JUL2000
2	Meena	25DEC1999

3.19.1 Specifying SAS Formats

Definition

Specifying SAS formats means assigning display formats to variables so that output is shown in a meaningful way.

Syntax

```
FORMAT variable format.;
```

3.19.2 Specifying SAS Formats

Common SAS Formats

Format	Use
`DATE9.`	Displays date as `07JUL2026`
`MMDDYY10.`	Displays date as `07/07/2026`
`DOLLAR8.2`	Displays currency
`COMMA10.`	Displays numbers with commas
`PERCENT8.2`	Displays percentages
`$CHARw.`	Displays character values

3.19.3 Specifying SAS Formats

A salary value `45000` can be displayed as `$45,000.00` using a dollar format.

Numerical Example

```
DATA salary;
INPUT empid salary taxrate;
FORMAT salary DOLLAR10.2 taxrate PERCENT8.2;
DATALINES;
101 45000 0.10
102 60000 0.15
;
RUN;

PROC PRINT DATA=salary;
RUN;
```

Output:

EmpID	Salary	Taxrate
101	$45,000.00	10.00%
102	$60,000.00	15.00%

3.19.4 Specifying SAS Formats: Example 1

CODE LOG RESULTS OUTPUT DATA

```
1  * SAS Format example by Viru;
2  DATA salary;
3  INPUT empid salary taxrate;
4  FORMAT salary DOLLAR10.2 taxrate PERCENT8.2;
5  DATALINES;
6  101 45000 0.10
7  102 60000 0.15
8  ;
9  RUN;
10
11 PROC PRINT DATA=salary;
12 RUN;
```

CODE LOG RESULTS OUTPUT DATA

Table of Contents

Obs	empid	salary	taxrate
1	101	$45,000.00	10.00%
2	102	$60,000.00	15.00%

CODE LOG RESULTS OUTPUT DATA

Table: WORK.SALARY | View: Column names

(none)

Columns

- Select all
- empid
- salary
- taxrate

Total rows: 2 Total columns: 3

	empid	salary	taxrate
1	101	$45,000.00	10.00%
2	102	$60,000.00	15.00%

3.A Summary

Topic	Main Keyword	Important Statement
Structure of data sets	Metadata	`PROC CONTENTS`
Testing program	Debugging	SAS log, `OPTIONS OBS=`
Combining data sets	Integration	`SET`, `MERGE`
One-to-one reading	Row-wise combining	Multiple `SET` statements
Concatenating	Vertical joining	`SET data1 data2;`
Match-merging	Key-based joining	`MERGE` with `BY`
Renaming variables	Name change	`RENAME`
Excluding unmatched	Matched only	`IF a AND b;`
Selecting matching	Inner join logic	`IN=` variables
DO loops	Repetition	`DO i=1 TO n;`
OUTPUT	Explicit writing	`OUTPUT;`
Nested loops	Loop inside loop	Total = outer × inner
DO UNTIL	Runs until true	Condition checked last
DO WHILE	Runs while true	Condition checked first
Arrays	Variable grouping	`ARRAY`
Formats	Display values	`FORMAT`
Informats	Read values	`INFORMAT`

3.B Concatenation vs Match-Merging

Basis	Concatenation	Match-Merging
SAS statement	`SET`	`MERGE`
Combination type	Vertical	Horizontal
BY variable required	No	Yes
Sorting required	No	Usually yes
Example	Monthly sales stacking	Employee salary merging

3.C DO WHILE vs DO UNTIL

Basis	DO WHILE	DO UNTIL
Condition checked	Before loop	After loop
Minimum execution	0 times	1 time
Continues when	Condition is true	Condition is false
Stops when	Condition becomes false	Condition becomes true

3.D Format vs Informat

Basis	Format	Informat
Purpose	Display data	Read data
Used with	Output	Input
Example	`DATE9.`	`MMDDYY10.`
Changes stored value?	No	Helps create stored value

3.E Sample Questions

1. Explain different methods for combining SAS data sets.
2. Differentiate between concatenation and match-merging with suitable examples.
3. Explain the use of `IN=` option for selecting and excluding observations.
4. Explain `DO WHILE` and `DO UNTIL` statements with examples.
5. Write a SAS program using nested `DO` loops.
6. Explain arrays in SAS with syntax and example.
7. Differentiate between SAS formats and informats.
8. Explain how to determine the structure and contents of a SAS data set.
9. Explain the role of `OUTPUT` statement inside `DO` loops.
10. Write a SAS program to merge two data sets and keep only matching observations.

LECTURE 06 TOPICS

Unit 3 Advanced Statistical Techniques in SAS

3.1	Determining the Structure and Contents of Data Sets
3.2	Testing Program
3.3	Methods for Combining SAS Data Sets
3.4	One-to-One Reading
3.5	Concatenating
3.6	Match-Merging
3.7	Renaming Variables
3.8	Excluding Unmatched Observations
3.9	Selecting Matching Observations
3.1O	Basics of DO Loops
3.11	Executing DO Loops
3.12	OUTPUT Statements
3.13	Nesting DO Loops
3.14	Iteratively Processing Observations from a Data Set
3.15	DO UNTIL Statement
3.16	DO WHILE Statement
3.17	Defining and Processing Arrays
3.18	Applying SAS Formats and Informats
3.19	Specifying SAS Formats

3.1O	Basics of DO Loops
3.11	Executing DO Loops
3.12	OUTPUT Statements
3.13	Nesting DO Loops
3.14	Iteratively Processing Observations from a Data Set
3.15	DO UNTIL Statement
3.16	DO WHILE Statement
3.17	Defining and Processing Arrays
3.18	Applying SAS Formats and Informats
3.19	Specifying SAS Formats

LECTURE 06

21CSE295T

Statistical Data Analytics And Integration

Doubts?

THANK YOU!

Virupakshan K

+91-9840847872 | VIRU@KART.LA