LECTURE 04

21CSE295T

Statistical Data Analytics And Integration

Unit 02: Data Analysis Using SAS

2.10	SAS Process Programs
2.11	Compilation Phase
2.12	Program Data Vector (PDV)
2.13	Descriptor Portion
2.14	Execution Phase
2.15	Data Portion
2.16	Debugging a DATA Step
2.17	Testing Programs
2.18	BY-Group Processing Definitions
2.19	Preprocessing Data
2.20	Determine Whether the Data Requires Preprocessing
2.21	Sorting Observations for BY-Group Processing
2.22	FIRST. and LAST. DATA Step Variables
2.23	Assignment Statements
2.24	Operators in SAS Expressions

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

Unit 02: Data Analysis Using SAS

2.1 Basic Reports
2.2 Selecting Variables
2.3 Identifying Observations
2.4 Subsetting the Data
2.5 Limiting the Observations
2.6 SORT Procedure
2.7 Generating Column Totals
2.8 Specifying Titles and Footnotes in Procedure Output
2.9 Assigning Descriptive Labels and Permanent Labels
2.25 PROC PRINT Procedure
2.26 WHERE Statement
2.27 Comparison Operators (EQ, NE, GT, LT, GE, LE)
2.28 AND / OR / CONTAINS

2.10 SAS Process Programs
2.11 Compilation Phase
2.12 Program Data Vector (PDV)
2.13 Descriptor Portion
2.14 Execution Phase
2.15 Data Portion
2.18 BY-Group Processing Definitions
2.19 Preprocessing Data
2.20 Determine Whether the Data Requires Preprocessing
2.21 Sorting Observations for BY-Group Processing
2.22 FIRST. and LAST. DATA Step Variables

2.16 Debugging a DATA Step
2.17 Testing Programs
2.23 Assignment Statements
2.24 Operators in SAS Expressions

Lecture 04 Topics

2.10	SAS Process Programs
2.11	Compilation Phase
2.12	Program Data Vector (PDV)
2.13	Descriptor Portion
2.14	Execution Phase
2.15	Data Portion
2.16	Debugging a DATA Step
2.17	Testing Programs
2.18	BY-Group Processing Definitions
2.19	Preprocessing Data
2.20	Determine Whether the Data Requires Preprocessing
2.21	Sorting Observations for BY-Group Processing
2.22	FIRST. and LAST. DATA Step Variables
2.23	Assignment Statements
2.24	Operators in SAS Expressions

2.10.1 SAS Process Programs

A SAS program is processed in steps. The two major types of SAS steps are:

1. DATA step
2. PROC step

DATA Step

Used to create or modify SAS datasets.

```
DATA newdata;
    SET olddata;
RUN;
```

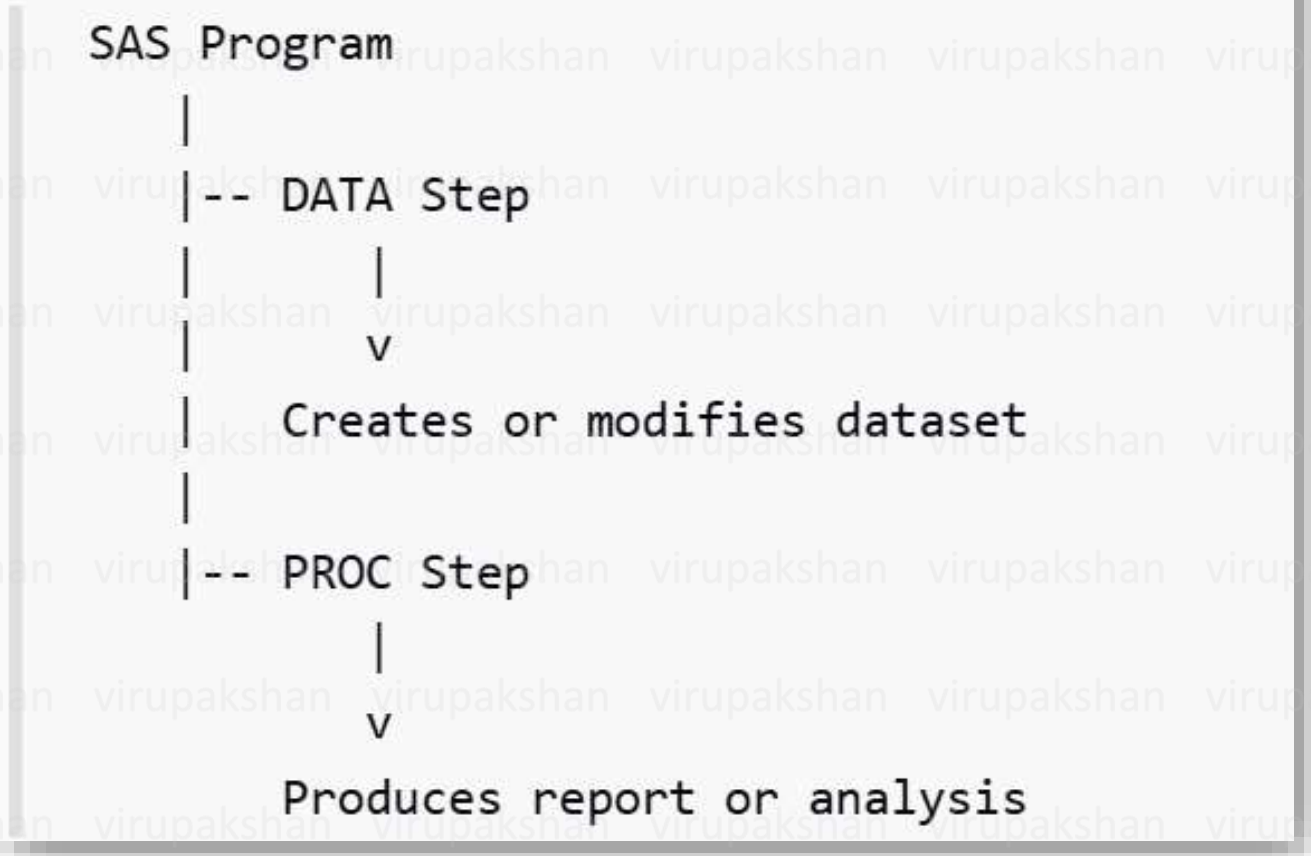
PROC Step

Used to analyze, print, sort, or summarize data.

```
PROC PRINT DATA=newdata;
RUN;
```

2.10.2 SAS Process Programs

Diagram: SAS Program Processing

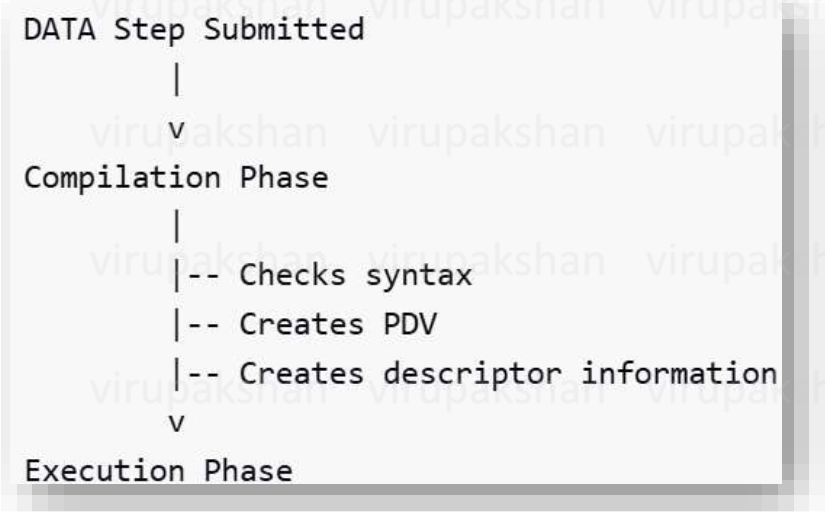


2.11 Compilation Phase

The **compilation phase** is the first phase of DATA step processing.

During compilation, SAS:

- Checks syntax errors.
- Creates the Program Data Vector, also called PDV.
- Creates the descriptor portion of the dataset.
- Identifies variable names, types, and lengths.



Example

```
DATA class;
   SET students;
   Grade = Marks / 10;
RUN;
```

During compilation, SAS identifies:

- ID
- Name
- Gender
- Marks
- Age
- Grade

2.12.1 Program Data Vector (PDV)

The **Program Data Vector**, or **PDV**, is a temporary memory area where SAS builds one observation at a time.

It contains:

- Variables from the input dataset.
- Newly created variables.
- Automatic variables.

Automatic Variables

Variable	Meaning
_N_	Number of times DATA step has executed
ERROR	Indicates whether an error occurred

2.12.2 Program Data Vector (PDV)

Example

```
DATA class;
   SET students;
   Grade = Marks / 10;
RUN;
```

PDV Concept

For the first observation:

ID	Name	Gender	Marks	Age	Grade	*N*	*ERROR*
101	Ravi	M	78	20	7.8	1	0

2.13.3 Descriptor Portion

The **descriptor portion** stores metadata about a SAS dataset.

Metadata means data about data.

It contains:

- Dataset name.
- Number of observations.
- Number of variables.
- Variable names.
- Variable types.
- Variable lengths.
- Labels.
- Formats.

Example

To view descriptor information:

```
PROC CONTENTS DATA=students;
RUN;
```

Output Concept

Variable	Type	Length	Label
ID	Numeric	8	Student ID
Name	Character	8	Student Name
Marks	Numeric	8	Exam Marks

2.14.4 Execution Phase

The **execution phase** is the second phase of DATA step processing.

During execution, SAS:

- Reads data one observation at a time.
- Executes statements.
- Calculates new variables.
- Writes observations to the output dataset.

Example

```
DATA pass_students;
    SET students;
    IF Marks >= 50;
RUN;
```

During execution, SAS reads each row and checks if `Marks >= 50`.

Execution Flow

```
Read observation 1
    |
Apply statements
    |
Write to output dataset
    |
Read observation 2
    |
Apply statements
    |
Write to output dataset
```

2.15.5 Data Portion

The **data portion** contains the actual data values stored in the SAS dataset.

A SAS dataset has two main parts:

```
SAS Dataset
    |
    |-- Descriptor Portion
    |       Metadata
    |
    |-- Data Portion
            Actual observations and values
```

Example Data Portion

ID	Name	Gender	Marks	Age
101	Ravi	M	78	20
102	Anu	F	85	19

2.16.1 Debugging a DATA Step

Debugging means finding and correcting errors in a SAS program.

Common SAS Errors

Error Type	Example
Syntax error	Missing semicolon
Logic error	Wrong condition
Data error	Invalid numeric value
Variable error	Misspelled variable name

Example of Error

```
DATA test;
    SET students
    Total = Marks + 10;
RUN;
```

Error:

Missing semicolon after `SET students.`

Correct Code

```
DATA test;
    SET students;
    Total = Marks + 10;
RUN;
```

2.16.2 Debugging a DATA Step: Useful Debugging Techniques

1. Check the SAS Log

The SAS log shows errors, warnings, and notes.

2. Use PUT Statement

```
DATA test;
    SET students;
    Grade = Marks / 10;
    PUT Name= Marks= Grade=;
RUN;
```

3. Use OBS= for Testing

```
DATA test;
    SET students(OBS=5);
RUN;
```

2.16.3a Debugging a DATA Step: PUT example

1. DATA Statement DATA test;

- Creates a new SAS dataset named **test**.
- The processed data will be stored in this dataset.

```
DATA test;
    SET students;
    Grade = Marks / 10;
    PUT Name= Marks= Grade=;
RUN;
```

2. SET Statement SET students;

- Reads observations from the existing dataset **students** one at a time.
- Each observation is loaded into the Program Data Vector (PDV) for processing.

Suppose the **students** dataset contains:

Name	Marks
Ravi	85
Anu	90
Kiran	75

2.16.3b Debugging a DATA Step : PUT example

3. Assignment Statement **Grade = Marks / 10;**

- Creates a new variable **Grade**.
- Calculates Grade by dividing Marks by 10.

```
DATA test;
    SET students;
    Grade = Marks / 10;
    PUT Name= Marks= Grade=;
RUN;
```

4. PUT Statement **PUT Name= Marks= Grade=;**

* Writes values to the SAS Log (not to the output window).
* The = sign tells SAS to print both the variable name and its value.

Example log output:

```
Name=Ravi Marks=85 Grade=8.5
Name=Anu Marks=90 Grade=9
Name=Kiran Marks=75 Grade=7.5
```

This is useful for:

- Debugging programs
- Checking calculations
- Monitoring data processing

5. RUN Statement **RUN;**

Executes the DATA step.

2.16.4 Debugging a DATA Step : OBS example

```
DATA test;
    SET students(OBS=5);
RUN;
```

This program creates a new dataset called **test** by copying only the **first 5 observations** from the dataset **students**.

1. DATA Statement **DATA test;**

- Creates a new SAS dataset named **test**.
- The selected observations will be stored in this dataset.

2. SET Statement with OBS= **SET students(OBS=5);**

- Reads data from the existing dataset **students**.
- The dataset option OBS=5 tells SAS to read **only the first 5 observations**.
- Any observations after the fifth one are ignored.

3. RUN Statement **RUN;** Executes the DATA step.

Obs	Name	Marks
1	Ravi	85
2	Anu	90
3	Kiran	75
4	Meena	88
5	John	79
6	Priya	92
7	Arjun	81

SET students(OBS=5);

Obs	Name	Marks
1	Ravi	85
2	Anu	90
3	Kiran	75
4	Meena	88
5	John	79

2.16.5 Debugging a DATA Step : OBS example

Why Use OBS=?

OBS= is useful when:

- Testing a program on a small sample of data.
- Working with very large datasets.
- Debugging code quickly.
- Creating sample datasets for demonstrations.

```
DATA test;
SET students(FIRSTOBS=3 OBS=5);
RUN;
```

Obs	Name	Marks
1	Ravi	85
2	Anu	90
3	Kiran	75
4	Meena	88
5	John	79
6	Priya	92
7	Arjun	81

Obs	Name	Marks
3	Kiran	75
4	Meena	88
5	John	79

2.17 Testing Programs

Testing programs means checking whether a SAS program works correctly before using it on the full dataset.

Testing Methods

Method	Purpose
Use small sample data	Saves time
Use `OBS=`	Limits observations
Use `PROC PRINT`	Checks output
Use `PROC CONTENTS`	Checks structure
Check SAS log	Finds errors

2.18 BY-Group Processing Definitions

BY-group processing means processing observations in groups based on the values of one or more variables.

Example

If students are grouped by gender:

Gender	Name	Marks
F	Anu	85
F	Meena	92
M	Ravi	78
M	John	67

Each gender forms a BY group.

Syntax

```
PROC PRINT DATA=students;
    BY Gender;
RUN;
```

Before using `BY`, the data must usually be sorted by that variable.

2.19 Preprocessing Data

Preprocessing means preparing data before analysis.

It may include:

- Sorting data.
- Removing missing values.
- Creating new variables.
- Correcting errors.
- Filtering observations.
- Formatting variables.
- Combining datasets.

Example

```
DATA clean_students;
    SET students;
    IF Marks NE .;
    Grade = Marks / 10;
RUN;
```

This removes **missing marks** and creates a new variable called **`Grade`**.

2.20 Determine Whether the Data Requires Preprocessing

Data requires preprocessing if it has:

- Missing values.
- Duplicate records.
- Incorrect data types.
- Outliers.
- Inconsistent categories.
- Unsorted observations.
- Invalid values.

Useful Procedures

Check Structure

```
PROC CONTENTS DATA=students;
RUN;
```

Check Data Values

```
PROC PRINT DATA=students;
RUN;
```

Check Frequency

```
PROC FREQ DATA=students;
    TABLES Gender;
RUN;
```

Check Summary Statistics

```
PROC MEANS DATA=students;
    VAR Marks Age;
RUN;
```

2.21 Sorting Observations for BY-Group Processing

Before using **BY-group** processing, **data must be sorted** by the BY variable.

Example:

```
PROC SORT DATA=students OUT=students_sorted;
    BY Gender;
RUN;

PROC PRINT DATA=students_sorted;
    BY Gender;
RUN;
```

Important Rule

If data is not sorted, SAS may produce an error:

```
ERROR: Data set is not sorted in ascending sequence.
```

2.22.1 FIRST. and LAST. DATA Step Variables

When using BY-group processing in a DATA step, SAS automatically creates temporary variables:

- `FIRST.variable`
- `LAST.variable`

These variables identify the first and last observation in each BY group.

2.22.2 FIRST. and LAST. DATA Step Variables

Example Dataset

Gender	Name	Marks
F	Anu	85
F	Meena	92
M	Ravi	78
M	John	67

Output Concept

Gender	TotalMarks
F	177
M	145

Example Code

```
PROC SORT DATA=students OUT=students_sorted;
    BY Gender;
RUN;

DATA gender_summary;
    SET students_sorted;
    BY Gender;

    IF FIRST.Gender THEN TotalMarks = 0;
    TotalMarks + Marks;

    IF LAST.Gender THEN OUTPUT;
RUN;
```

2.22.3 FIRST. and LAST. DATA Step Variables

Explanation

- `FIRST.Gender` is 1 for the first observation of each gender.
- `LAST.Gender` is 1 for the last observation of each gender.
- `TotalMarks + Marks;` is a sum statement that retains the value automatically.

Example Dataset

Gender	Name	Marks
F	Anu	85
F	Meena	92
M	Ravi	78
M	John	67

Diagram

```
Gender = F
   |
   |-- FIRST.Gender = 1 for Anu
   |-- LAST.Gender  = 1 for Meena

Gender = M
   |
   |-- FIRST.Gender = 1 for Ravi
   |-- LAST.Gender  = 1 for John
```

Output Concept

Gender	TotalMarks
F	177
M	145

2.22.4a FIRST. and LAST. DATA Step Variables

```
PROC SORT DATA=students OUT=students_sorted;
    BY Gender;
RUN;

DATA gender_summary;
    SET students_sorted;
    BY Gender;

    IF FIRST.Gender THEN TotalMarks = 0;
    TotalMarks + Marks;

    IF LAST.Gender THEN OUTPUT;
RUN;
```

What the program does

This program calculates the **total marks for each Gender group** (for example, M and F).

It works in two stages:

1. Sort the data by Gender.
2. Process each Gender group and accumulate marks.

2.22.4b FIRST. and LAST. DATA Step Variable

Step 1: Sort the data


```
PROC SORT DATA=students OUT=students_sorted;
    BY Gender;
```

Statement	Meaning
`PROC SORT`	Invokes the SAS sorting procedure.
`DATA=students`	The input dataset is `students`.
`OUT=students_sorted`	The sorted result will be stored in a new dataset called `students_sorted`.
`BY Gender;`	Sort the observations by the variable `Gender`.

Why is sorting required?

The next DATA step uses **`BY Gender`** processing.

In SAS, BY-group processing requires the data to be sorted by the BY variable first.

2.22.4c FIRST. and LAST. DATA Step Variable: Step 1: Sort the data

Before Sorting by Gender

Gender	Name	Marks
F	Anu	85
M	John	67
M	Ravi	78
F	Meena	92

After Sorting by Gender

Gender	Name	Marks
F	Anu	85
F	Meena	92
M	Ravi	78
M	John	67

```
PROC SORT DATA=students OUT=students_sorted;
    BY Gender;
```

2.22.4d FIRST. and LAST. DATA Step Variab

Step 2: Create group summaries

1. Create a new dataset

SAS will create a dataset named **gender_summary.**

2. Read the sorted data

Reads observations one by one from **students_sorted.**

3. Enable BY-group processing

This tells SAS to process observations grouped by **Gender**.
SAS automatically creates two temporary variables for each BY variable:

Temporary Variable	Meaning
FIRST.Gender	1 for the first observation of a Gender group.
LAST.Gender	1 for the last observation of a Gender group.

4. Initialize the total at the start of each group

When SAS reaches the first record of a Gender group, it resets **TotalMarks** to 0.

Example for Gender = F:

Name	FIRST.Gender	Action
Anu	1	TotalMarks = 0
Meena	0	No reset

5. Add marks to the running total

This is a sum statement.

Name	Marks	TotalMarks
Anu	90	90
Meena	88	178

6. Output only the last record of each group

Only when SAS reaches the last observation of a Gender group does it write a record to **gender_summary**.

Example for Gender = F:

Name	LAST.Gender	Action
Anu	0	No Output
Meena	1	Output record

2.22.5 FIRST. and LAST. DATA Step Variables

What does the output look like?

Using the example data:

Gender	Name	Marks
F	Anu	85
F	Meena	92
M	Ravi	78
M	John	67

The resulting dataset `gender_summary` will contain:

Gender	Name	Marks	TotalMarks
F	Meena	92	177
M	John	67	145

Notice that only the last observation from each gender group is kept, but `TotalMarks` contains the sum for the entire group.

Important note:

The output still contains variables such as `Name` and `Marks` from the last observation in each group. If you want a cleaner summary dataset, you might later use **`KEEP`** or **`DROP`** statements to retain only `Gender` and `TotalMarks`.

2.23 Assignment Statements

An **assignment statement** creates or changes the value of a variable.

Syntax:

```
variable = expression;
```

Examples:

```
Grade = Marks / 10;
Total = Marks + Bonus;
Status = "Pass";
```

Example Program:

```
DATA students_new;
    SET students;
    Bonus = 5;
    FinalMarks = Marks + Bonus;
RUN;
```

2.24 Operators in SAS Expressions

Operators are symbols or keywords used to perform calculations or comparisons.

Types of Operators

Type	Operators
Arithmetic	+ , - , * , / , **
Comparison	= , ^= , > , < , >= , <=
Mnemonic comparison	EQ , NE , GT , LT , GE , LE
Logical	AND , OR , NOT
Character	`
Special	IN , CONTAINS

Arithmetic Example

```
FinalMarks = Marks + 5;
Average = Total / 5;
Square = X ** 2;
```

Character Concatenation Example

```
FullName = FirstName || " " || LastName;
```

Lecture 04 Topics

2.10	SAS Process Programs
2.11	Compilation Phase
2.12	Program Data Vector (PDV)
2.13	Descriptor Portion
2.14	Execution Phase
2.15	Data Portion
2.16	Debugging a DATA Step
2.17	Testing Programs
2.18	BY-Group Processing Definitions
2.19	Preprocessing Data
2.20	Determine Whether the Data Requires Preprocessing
2.21	Sorting Observations for BY-Group Processing
2.22	FIRST. and LAST. DATA Step Variables
2.23	Assignment Statements
2.24	Operators in SAS Expressions

LECTURE 04

21CSE295T

Statistical Data Analytics And Integration

Doubts?

THANK YOU!

Virupakshan K

+91-9840847872 | VIRU@KART.LA