PRACTICAL JOURNAL

in

SAP ANALYTICS

Submitted by

KSMSCIT005 HITESH BHANUSHALI

for the award of the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

PART – II

DEPARTMENT OF INFORMATION TECHNOLOGY
KISHINCHAND CHELLARAM COLLEGE
(Affiliated to University of HSNCU)
MUMBAI,400020
MAHARASHTRA
2024-25

SUBJECT CODE – BIT610D

SAP ANALYTICS

KISHINCHAND CHELLARAM COLLEGE

CHURCHGATE, MUMBAI – 400 020.

DEPARTMENT OF INFORMATION TECHNOLOGY

M.SC.I.T PART- II

CERTIFICATE

This is to certify that the Practical conducted by Mr. **HITESH VERSHI BHANUSHALI** for M.Sc. (IT) Part- II Semester- III, Seat No: **KSMSCIT005** at Kishinchand Chellaram College in partial fulfillment for the MASTERS OF SCIENCE (INFORMATION TECHNOLOGY). Degree Examination for Semester III has been periodically examined and signed, and the course of term work has been satisfactorily carried out for the year 2024 - 2025. This Practical journal had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Signature	**Signature**	**Signature**
Lecturer-In-Charge	**External Examiner**	**Course Coordination**
Guided By	**Examined By**	**Certified By**

College Stamp

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-1 Semester II

INDEX

Sr. No.	TITLE	Date	Page No.	Signature
	PRACTICALS			
1	Create an ABAP program that declares 2 variables and displays the sum of 2 variables in output using write statement.	18/08/24	1	
2	Write an ABAP program that uses an IF statement to check if a number is positive, negative, or zero, and displays the result.	25/08/24	2	
3	Define a database table in the ABAP Dictionary with fields for ID, NAME, and AGE. Include data elements and domains for these fields.	15/09/24	3	
4	Create an ABAP program that includes a subroutine to calculate the factorial of a given number. Call this subroutine from the main program.	22/09/24	6	
5	Create an ALV (ABAP List Viewer) report that displays employee data from a custom table. Use ALV to provide sorting and filtering capabilities	29/09/24	7	
6	Linear Define a simple ABAP class with attributes for Name and Age. Include methods to set and get these attributes. Instantiate the class and demonstrate its usage.	29/09/24	8	
7	Develop an ABAP program that performs a join between two tables and displays the results. Ensure that the join is based on a common key.	06/10/24	10	
8	Describe and execute a debugging session for an ABAP program that is not producing the expected output. Use breakpoints and watchpoints to identify the issue	13/10/24	12	
9	Explain the key differences between SAP ECC and S/4HANA in terms of data model and performance. Include examples of how ABAP programming might need to adapt.	13/10/24	14	

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 1

Aim: Create an ABAP program that declares 2 variables and displays the sum of 2 variables in output using write statement.

Theory:

In ABAP, variables are data objects that store values of different types such as integers, characters, or dates. Variables are declared with a specific type (e.g., integer) and assigned an initial value if needed. To perform operations like addition, ABAP uses operators such as +, -, , and /. In this example, two variables are declared and initialized with integer values, then summed, and the result is displayed. The WRITE statement is typically used in ABAP to output data to the screen or log. ABAP is primarily used for developing business applications, and handling numerical data operations like addition is a common task in financial calculations, inventory management, and various business processes within the SAP environment. Summing variables and displaying results can be essential for reports or validating data in real-time applications.

Code:

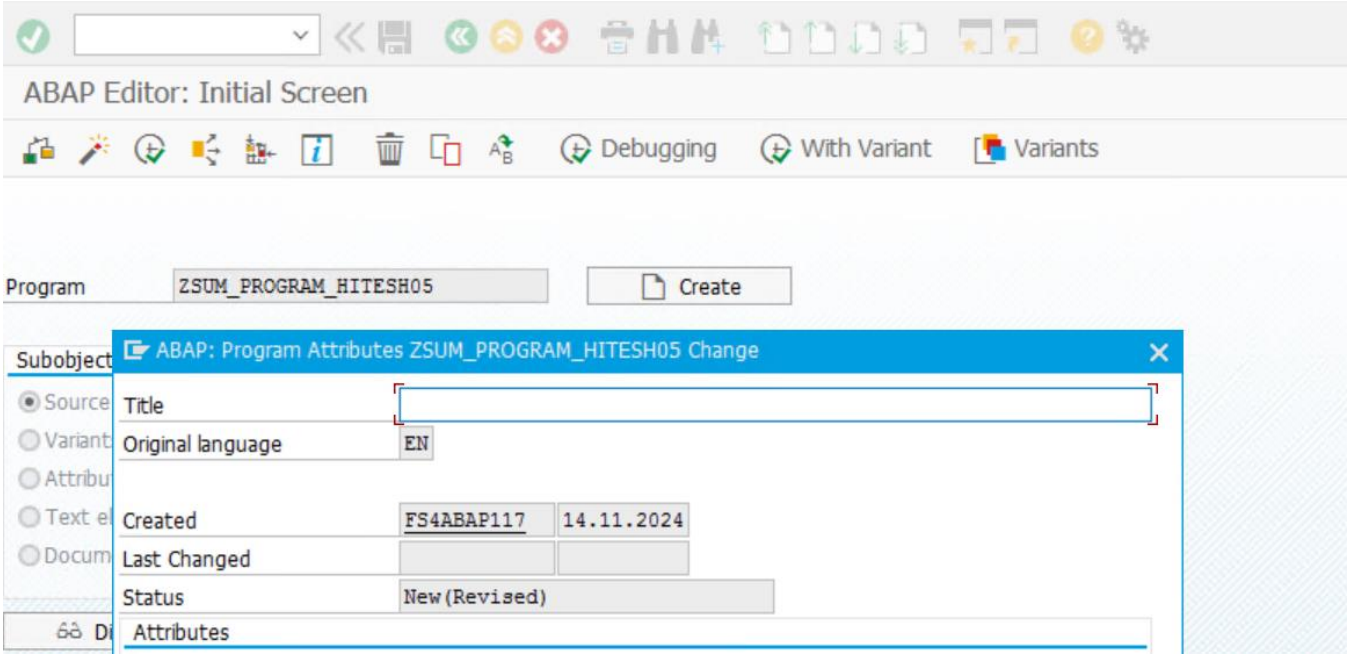
```
REPORT ZADD_TWO_VARIABLES.

 Declare two variables

DATA: num1 TYPE I VALUE 10,

          num2 TYPE I VALUE 20,

         result TYPE I.

  Calculate the sum

result = num1 + num2.

  Display the result

WRITE: ‘KSMSCIT005 HITESH BHANUSHALI’.

WRITE: / 'The sum of', num1, 'and', num2, 'is', result.
```

Steps to Perform

1. Log into the SAP System: Open the SAP GUI and log in to your SAP system.

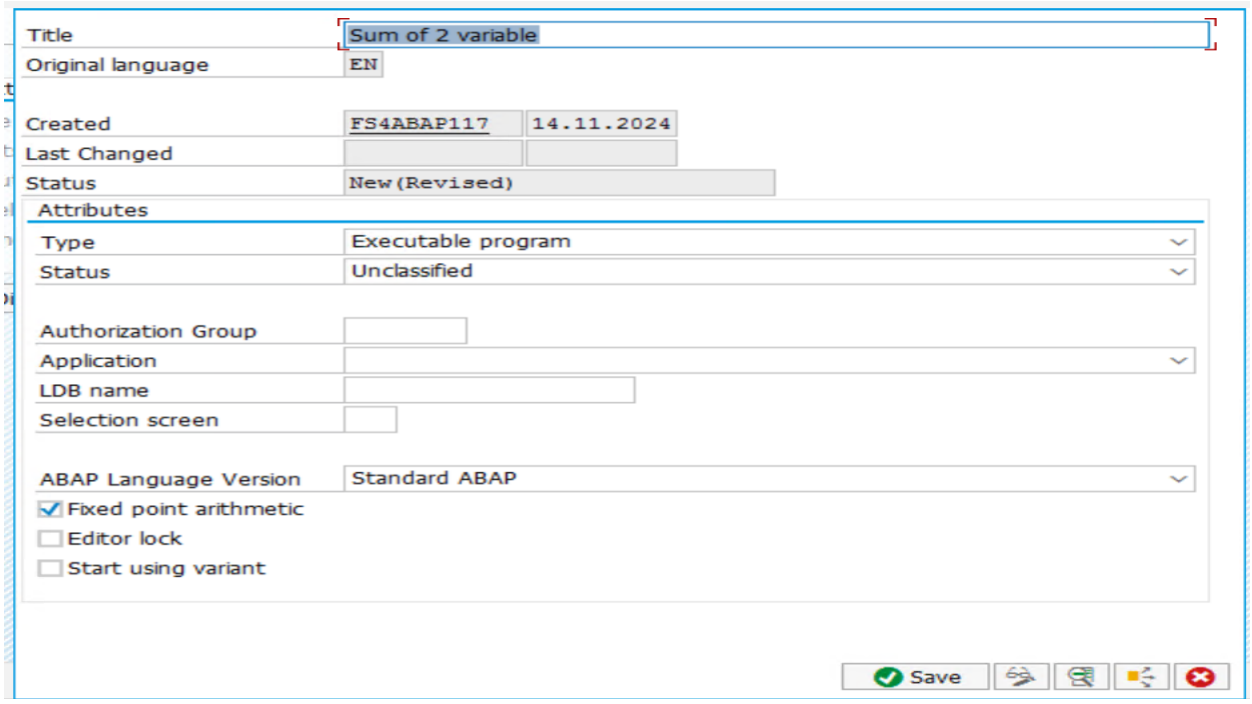
2. Access ABAP Editor:

- Go to the transaction code **SE38** or **SE80**.
- In SE38, type a program name (e.g., ZSUM_PROGRAM_HITESH05) and click on Create.



3. Create the Program:

- Enter a title for the program, such as "Sum of Two Variables."
- Set the type to **Executable Program**.
- Click **Save** and choose a package or select **Local Object** if it's a temporary program.



Now Add Package as **$TMP + Enter** and Save

4. Write the ABAP Code:

- In the ABAP editor, enter the following code:

ABAP Editor: Change Report ZSUM_PROGRAM_HITESH05

Report ZSUM_PROGRAM_HITESH05 inactive (revised)

```
*&
*&---------------------------------------------------------------------
REPORT ZSUM_PROGRAM_HITESH05.

DATA: VAR1 TYPE I VALUE 43,
      VAR2 TYPE I VALUE 51,
      SUM TYPE I.
SUM = VAR1 + VAR2.

WRITE: /'The sum of ', var1, 'and', var2, 'is', sum.
WRITE: 'Hitesh Bhanushali 05'.
```

5. Save and Check Syntax:

- Click Save.
- Click Check (or press Ctrl + F2) to ensure the code is free of errors.

6. Activate the Program:

- Click Activate (or press Ctrl + F3) to activate the program.

7. Run the Program:

- Click Execute (or press F8) to run the program.

OUTPUT:

Sum of 2 variable

```
Sum of 2 variable

The sum of          43  and         51  is         94  Hitesh Bhanushali 05
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 2

Aim: Write an ABAP program that uses an IF statement to check if a number is positive, negative, or zero, and displays the result.

Theory:

Conditional statements like IF are fundamental in ABAP for controlling program flow. The IF statement checks conditions and executes specific code segments based on whether these conditions are true or false. In this exercise, the program evaluates a variable to determine if it is positive, negative, or zero. Using IF, ELSEIF, and ELSE conditions, the program can control the flow of execution based on the value of the variable. This logic is essential in various business scenarios, such as validating input, categorizing data, or enforcing business rules. For example, the ability to check if a number is positive or negative is useful in financial transactions, inventory management, and error handling. This form of conditional logic allows ABAP programs to be more dynamic and responsive to different types of input and conditions within SAP systems.

Code:

```
REPORT ZCHECK_NUMBER_GRP8.

//Declare a variable to hold the input number

DATA: number TYPE I VALUE -5.

WRITE: / 'KSMSCIT005 HITESH BHANUSHALI'.

//Check if the number is positive, negative, or zero

IF number > 0.

 WRITE: / 'The number', number, 'is positive.'.

ELSEIF number < 0.

 WRITE: / 'The number', number, 'is negative.'.

ELSE.

 WRITE: / 'The number is zero.'.

ENDIF.
```

Steps to Perform

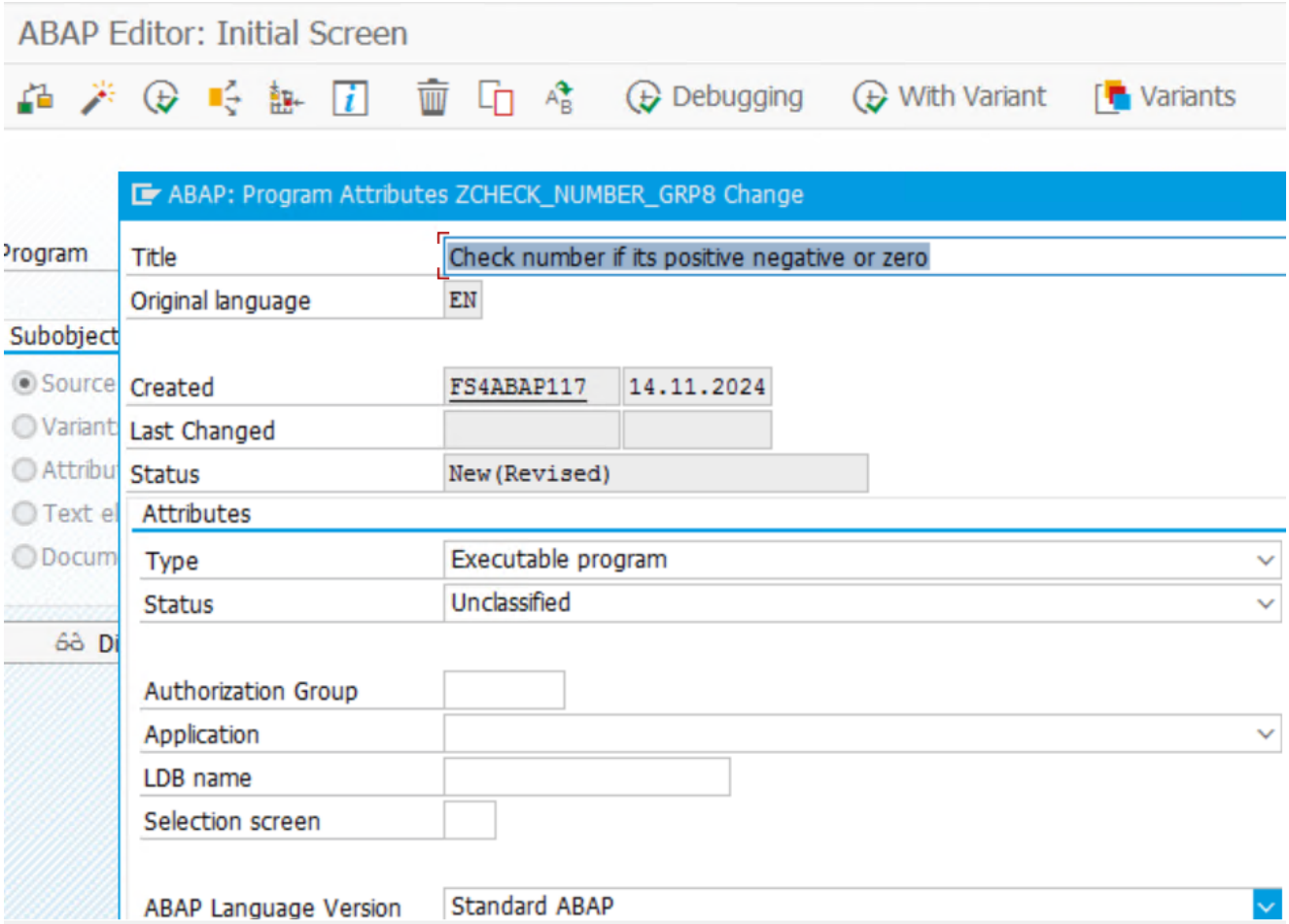
1. Log into the SAP System: Open the SAP GUI and log in to your SAP system.

2. Access ABAP Editor:

- Go to the transaction code SE38 or SE80.
- In **SE38**, type a program name (e.g., ZCHECK_NUMBER_GRP8) and click on Create

3. Create the Program:

- Enter a title for the program, such as "**Check number if its positive negative or zero**"
- Set the type to **Executable Program**.
- Click Save and choose a package or select Local Object if it's a temporary program.



4. Write the ABAP Code:

- In the ABAP editor, enter the following code:

ABAP Editor: Change Report ZCHECK_NUMBER_GRP8

Pattern Pretty Printer

Report ZCHECK_NUMBER_GRP8 inactive (revised)

```
*&---------------------------------------------------------------------*
*& Report ZCHECK_NUMBER_GRP8
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
REPORT ZCHECK_NUMBER_GRP8.
DATA: number TYPE I VALUE -5.
WRITE: / 'KSMSCIT005 HITESH BHANUSHALI'.
*& Check if the number is positive, negative, or zero
IF number > 0.
 WRITE: / 'The number', number, 'is positive.'.
ELSEIF number < 0.
 WRITE: / 'The number', number, 'is negative.'.
ELSE.
 WRITE: / 'The number is zero.'.
ENDIF.
```

OUTPUT:

Check number if its positive negative or zero

```
Check number if its positive negative or zero

KSMSCIT005 HITESH BHANUSHALI
The number          5- is negative.
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 3

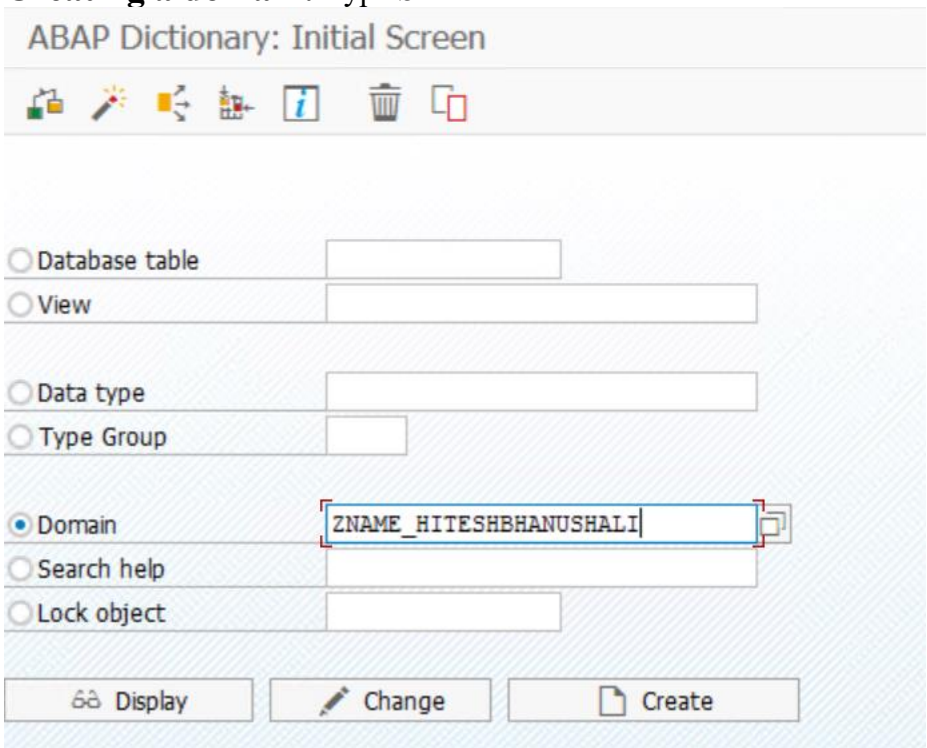
Aim: Define a database table in the ABAP Dictionary with fields for ID, NAME, and AGE. Include data elements and domains for these fields.

Theory:

Database tables in ABAP are defined within the ABAP Dictionary using transaction SE11. Each table contains fields (columns) that represent data elements with specific attributes, such as data type and length. Fields are created using data elements and domains, which define their technical characteristics and value ranges. For example, an employee table could have an ID (primary key), NAME (character string), and AGE (numeric). In SAP systems, the ABAP Dictionary centralizes database table definitions to ensure consistency, integrity, and efficiency. Database tables are essential for storing and retrieving data in SAP applications. By defining fields with appropriate data types and constraints, developers can ensure data accuracy and alignment with business requirements. The ABAP Dictionary also supports features like foreign keys, indexes, and views, which enhance database performance and data relationships within the SAP environment.

Steps:

Creating a domain: Type **SE11**



Dictionary: Change Domain

Domain: ZNAME_HITESHBHANUSHALI New(Revised)

Short Description: Student Name

Properties | Definition | Value Range

Format

Data Type: CHAR Character String

No. Characters: 15

Decimal Places: 0

Output Characteristics

Output Length: 15

Routine:

Sign

Case-sensitive

Create Object Directory Entry

Object: R3TR DOMA ZNAME_HITESHBHANUSHALI

Attributes

Package: $TMP

Person Responsible: FS4ABAP117

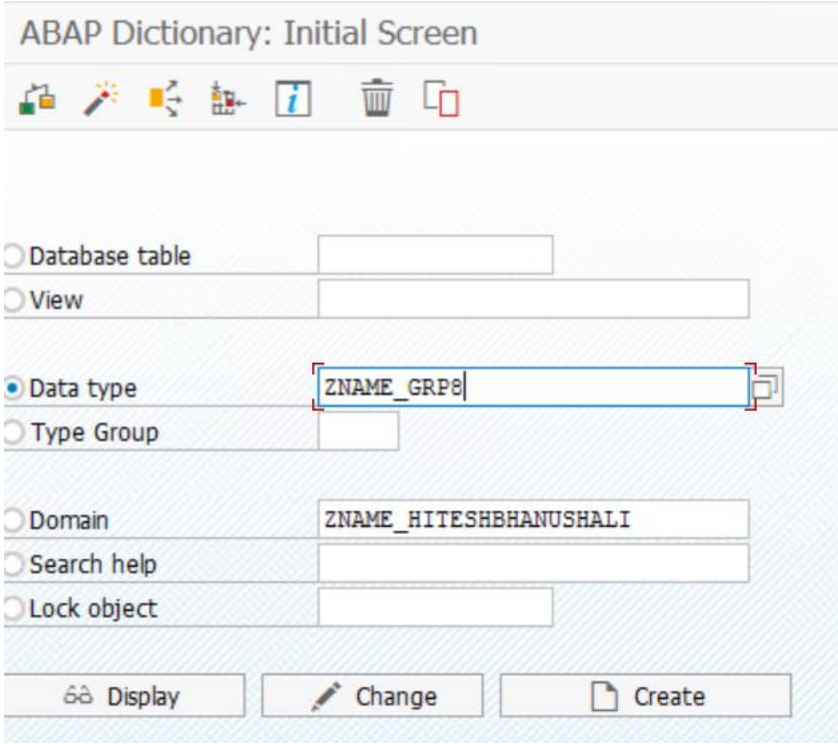
Original System: A4H

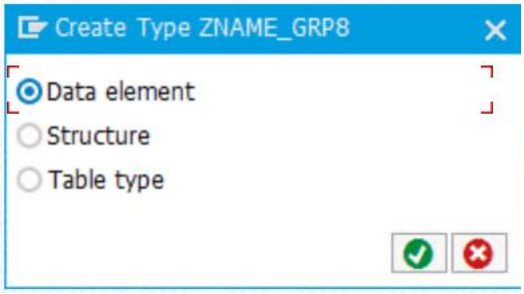
Original language: EN English

Created On:

Local Object | Lock Overview

Creating a data element:


ABAP Dictionary: Initial Screen
Database table
View
Data type
ZNAME_GRP8
Type Group
Domain
ZNAME_HITESHBHANUSHALI
Search help
Lock object
Display
Change
Create


Create Type ZNAME_GRP8
Data element
Structure
Table type

Dictionary: Change Data Element

Documentation Supplementary Docum

Data element: ZNAME_GRP8 New(Revised)

Short Description: STUDENT NAME

Attributes | Data Type | Further Characteristics | Field Label

Elementary Type

Domain: ZNAME_HITESHBHANUSHALI Student Name

Data Type: CHAR Character String

Length: 15

Built-in type

Data Type:

Length: 0

Reference Type

Referenced Type:

Reference to built-in type

Data Type:

Length: 0

Dictionary: Change Data Element

Documentation Supplementary Documenta

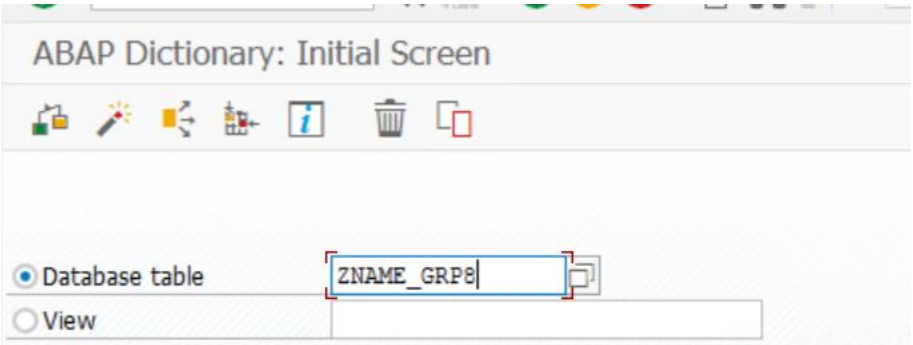
Data element: ZNAME_GRP8 New(Revised)

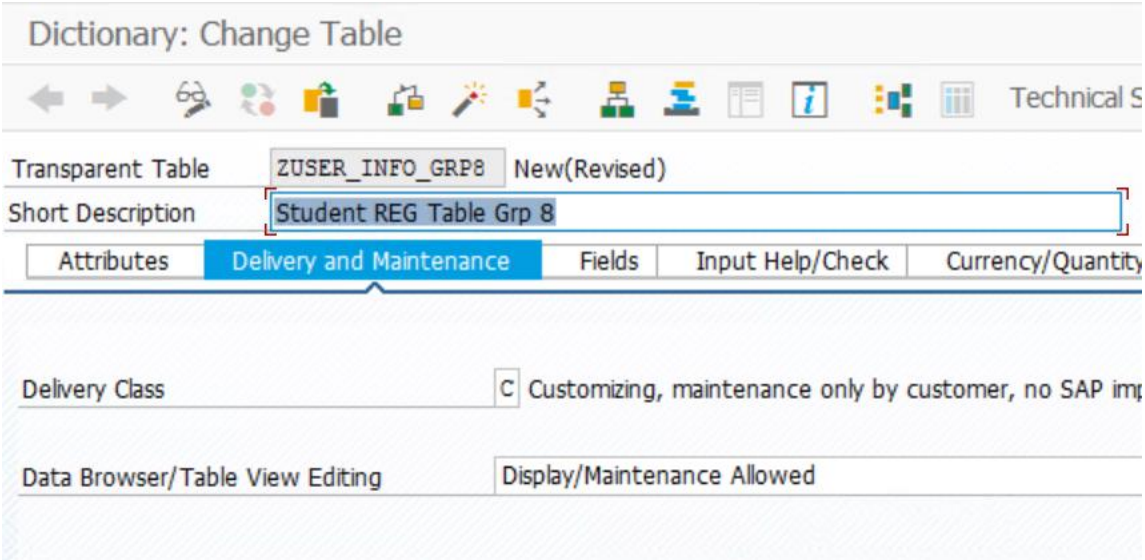
Short Description: STUDENT NAME

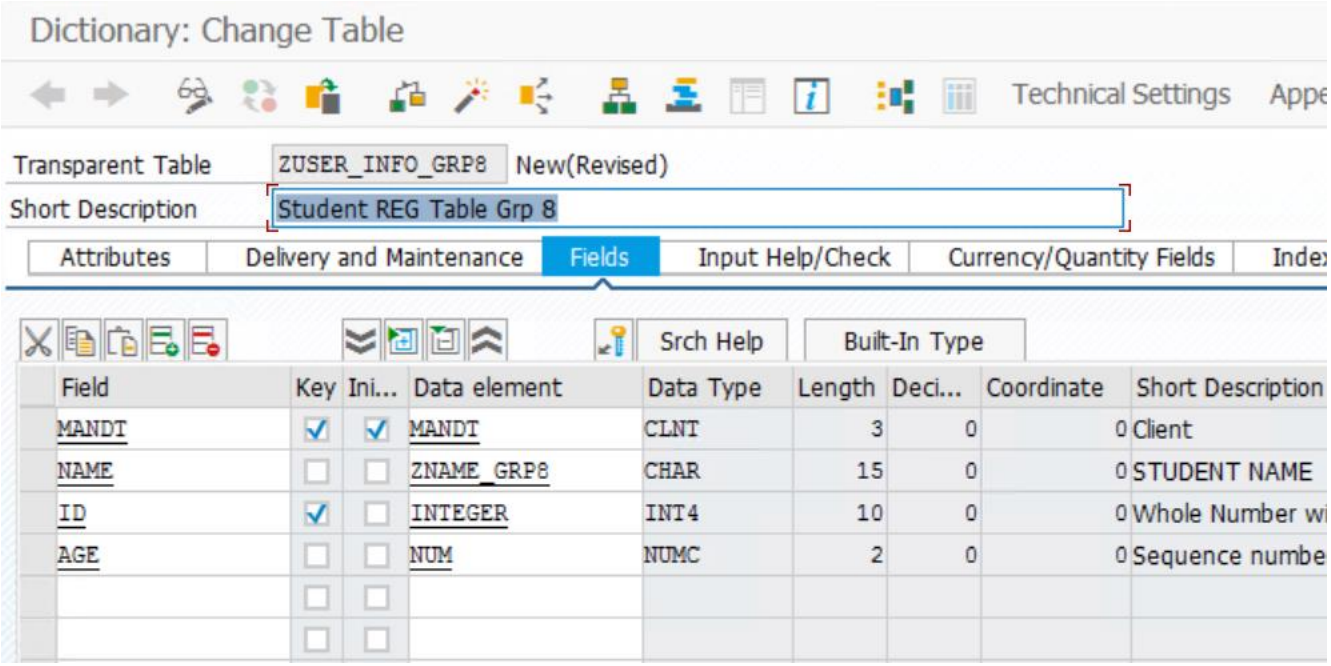
Attributes | Data Type | Further Characteristics | Field Label

	Length	Field Label
Short		STUD NAME
Medium		STUDENT NAME
Long		STUDENT FULL NAME
Heading		STUDENT FULL NAME FOR REGISTRATION

Creating a database:


ABAP Dictionary: Initial Screen
Database table
ZNAME_GRP8
View


Dictionary: Change Table
Technical S
Transparent Table
ZUSER_INFO_GRP8
New(Revised)
Short Description
Student REG Table Grp 8
Attributes
Delivery and Maintenance
Fields
Input Help/Check
Currency/Quantity
Delivery Class
C
Customizing, maintenance only by customer, no SAP imp
Data Browser/Table View Editing
Display/Maintenance Allowed


Dictionary: Change Table
Technical Settings
Appe
Transparent Table
ZUSER_INFO_GRP8
New(Revised)
Short Description
Student REG Table Grp 8
Attributes
Delivery and Maintenance
Fields
Input Help/Check
Currency/Quantity Fields
Index
Srch Help
Built-In Type
Field
Key
Ini...
Data element
Data Type
Length
Deci...
Coordinate
Short Description
MANDT
MANDT
CLNT
3
0
0
Client
NAME
ZNAME_GRP8
CHAR
15
0
0
STUDENT NAME
ID
INTEGER
INT4
10
0
0
Whole Number wi
AGE
NUM
NUMC
2
0
0
Sequence numbe

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 4

Aim: Create an ABAP program that includes a subroutine to calculate the factorial of a given number.

Call this subroutine from the main program.

Theory:

Subroutines in ABAP, defined with the FORM and ENDFORM keywords, are reusable blocks of code that can be called from multiple places within a program. Subroutines are typically used to perform tasks that are repeated or logically grouped, such as calculations, data formatting, or validations. In this example, a subroutine calculates the factorial of a given number by multiplying all integers up to that number. Using a subroutine helps to modularize code, making it easier to read, test, and maintain. Subroutines also enable code reusability, allowing developers to implement functions once and call them as needed. Factorial calculations are common in statistical analysis and other mathematical applications, though they are often used for demonstration purposes in programming. By defining and calling subroutines, ABAP developers can create structured programs that are organized and more efficient in business applications.

Code:

REPORT zfactorial_calc.

Declaration of variables

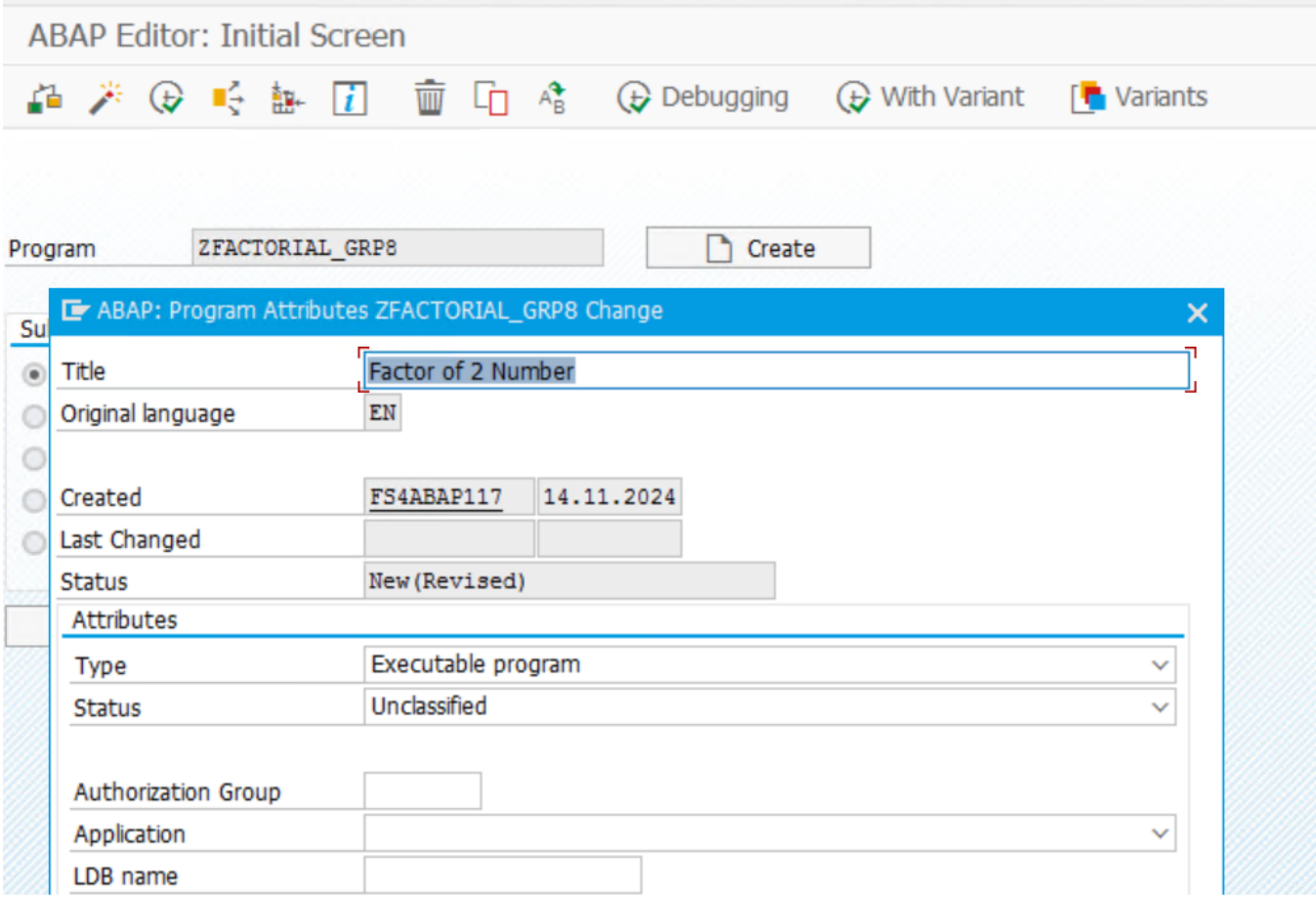
```
REPORT ZFACTORIAL_GRP8.

DATA: number TYPE i VALUE 5,
      factorial TYPE i VALUE 1.
**Subroutine to calculate factorial
FORM calculate_factorial USING num TYPE i
                 CHANGING fact TYPE i.
 DATA i TYPE i.
 fact = 1.
 DO num TIMES.
 fact = fact * sy-index.
 ENDDO.
ENDFORM.
**Main program logic
START-OF-SELECTION.
 WRITE:/ 'KSMSCIT005 HITESH BHANUSHALI '.
 PERFORM calculate_factorial USING number CHANGING factorial.
 WRITE: / 'Factorial of', number, 'is', factorial.
```

Steps to Perform

1. Log into the SAP System: Open the SAP GUI and log in to your SAP system.

2. Access ABAP Editor:

- Go to the transaction code SE38 or SE80.
- In SE38, type a program name (e.g., ZFACTORIAL-GEP8) and click on Create.



ABAP Editor: Change Report ZFACTORIAL_GRP8

Report ZFACTORIAL_GRP8 Inactive

```
*&---------------------------------------------------------------------
REPORT ZFACTORIAL_GRP8.

DATA: number TYPE i VALUE 5,
      factorial TYPE i VALUE 1.
**Subroutine to calculate factorial
FORM calculate_factorial USING num TYPE i
                         CHANGING fact TYPE i.
 DATA i TYPE i.
 fact = 1.
 DO num TIMES.
 fact = fact * sy-index.
 ENDDO.
ENDFORM.
**Main program logic
START-OF-SELECTION.
 WRITE:/ 'KSMSCIT005 HITESH BHANUSHALI '.
 PERFORM calculate_factorial USING number CHANGING factorial.
 WRITE: / 'Factorial of', number, 'is', factorial.
```

OUTPUT:

Factor of 2 Number

```
Factor of 2 Number

KSMSCIT005 HITESH BHANUSHALI
Factorial of          5  is        120
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 5

Aim: Create an ALV (ABAP List Viewer) report that displays employee data from a custom table. Use ALV to provide sorting and filtering capabilities.

Theory:

The ABAP List Viewer (ALV) is a powerful tool in SAP that enables users to display data in a structured, interactive format. ALV provides a variety of features, such as sorting, filtering, and exporting data, which enhances user experience and data analysis. ALV reports are often used for displaying data from database tables in a tabular form, allowing end-users to manipulate the view without altering the underlying data. For instance, an ALV report displaying employee data might include columns for ID, name, and department, with options to sort by these columns or apply filters to narrow results. ALV reporting improves data visibility and is widely used in SAP for creating business intelligence and operational reports. By leveraging ALV's features, developers can provide users with tools to analyze data effectively, making it an essential component of many SAP applications.

Code:

```
REPORT ZEMP_GRP8.
**Internal table to hold employee data
DATA: lt_STUDENTS TYPE TABLE OF ZEMP_GRP2.
**Fetch data from database
SELECT FROM ZEMP_GRP2 INTO TABLE lt_STUDENTS.
**ALV display function module
CALL FUNCTION 'REUSE_ALV_LIST_DISPLAY'
        EXPORTING
                i_structure_name = ' ZEMP_GRP8'
        TABLES
                t_outtab = lt_STUDENTS
        EXCEPTIONS
                program_error = 1
                others = 2.
IF sy-subrc <> 0.
        WRITE: / 'Error displaying ALV report'.
ENDIF.
```

Steps to Perform

1. Log into the SAP System: Open the SAP GUI and log in to your SAP system.
2. Access ABAP Editor:
   - Go to the transaction code SE38 or SE80.
   - In SE38, type a program name (e.g., ZEMP_GRP2) and click on Create.

If you need, you can create or use the existing table

ABAP Editor: Change Report ZEMP_GRP8

Report ZEMP_GRP8 Active

```
*&---------------------------------------------------------------------
REPORT ZEMP_GRP8.

**Internal table to hold employee data
DATA: lt_STUDENTS TYPE TABLE OF ZEMP_GRP2.
**Fetch data from database
SELECT * FROM ZEMP_GRP2 INTO TABLE lt_STUDENTS.
**ALV display function module
CALL FUNCTION 'REUSE_ALV_LIST_DISPLAY'
EXPORTING
i_structure_name = ' ZEMP_GRP8'
TABLES
t_outtab = lt_STUDENTS
EXCEPTIONS
program_error = 1
others = 2.
IF sy-subrc <> 0.
WRITE: / 'Error displaying ALV report'.
ENDIF.
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 6

Aim: Define a simple ABAP class with attributes for Name and Age. Include methods to set and get these attributes. Instantiate the class and demonstrate its usage.

Theory:

ABAP Objects, SAP's object-oriented extension of ABAP, allows developers to define classes with attributes and methods. A class in ABAP serves as a blueprint for creating objects, with attributes representing data and methods defining behavior. In this example, a simple class has attributes for Name and Age and methods to set and get these values. Object-oriented programming (OOP) in ABAP promotes modularity, reusability, and maintainability by organizing code into logical structures. Classes encapsulate data and functions, enhancing code organization and reducing redundancy. Creating instances (objects) of a class with methods to manipulate data supports scalable design. OOP is widely used in SAP, especially in complex applications where modular design and encapsulation improve code management. In real-world SAP applications, object-oriented ABAP is essential for developing custom business objects, such as customer or order classes, with clearly defined attributes and operations.

Code:

```
REPORT ZGETSET_GRP8.

CLASS ZCL_PERSON DEFINITION.

 PUBLIC SECTION.

 DATA: name TYPE STRING,

 age TYPE I.

METHODS: set_name IMPORTING i_name TYPE STRING,

 set_age IMPORTING i_age TYPE I,

 get_name RETURNING VALUE(r_name) TYPE STRING,

 get_age RETURNING VALUE(r_age) TYPE I.

ENDCLASS.

CLASS ZCL_PERSON IMPLEMENTATION.

 METHOD set_name.

 name = i_name.

 ENDMETHOD.

METHOD set_age.

 age = i_age.
```

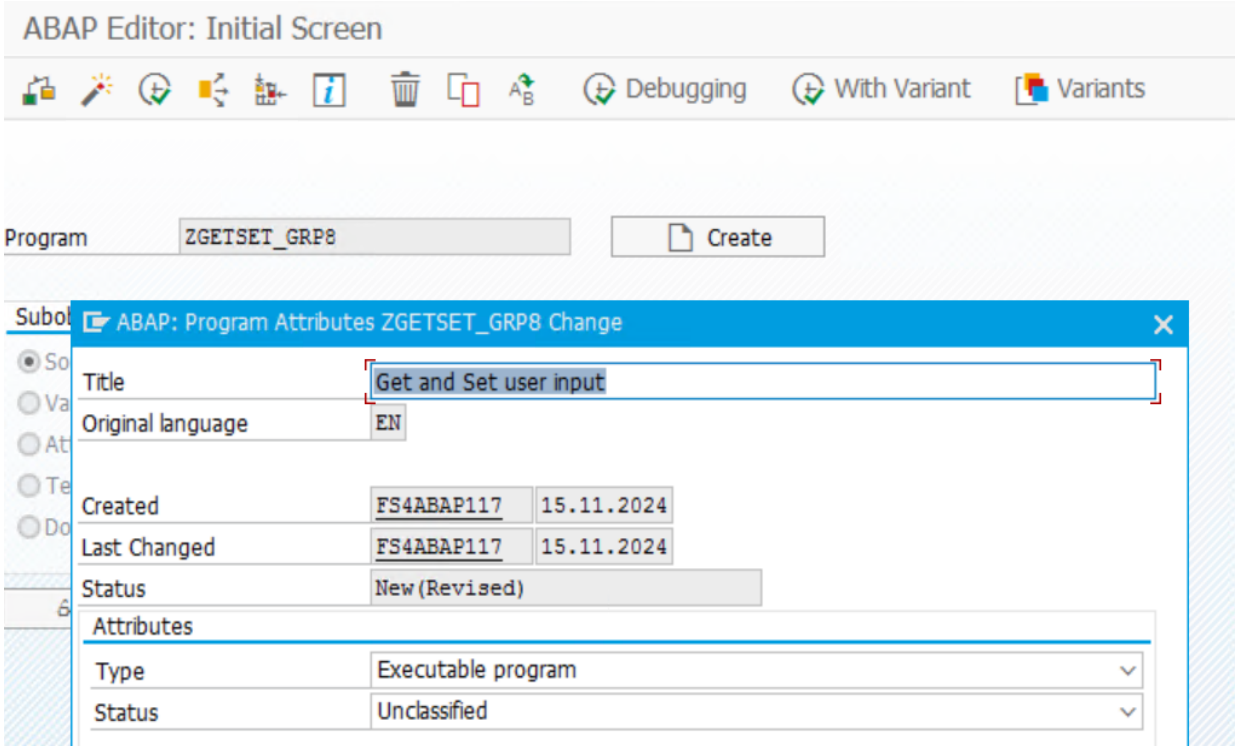
```
 ENDMETHOD.
METHOD get_name.
 r_name = name.
 ENDMETHOD.
METHOD get_age.
 r_age = age.
 ENDMETHOD.
ENDCLASS.
START-OF-SELECTION.
 DATA: person TYPE REF TO ZCL_PERSON.
CREATE OBJECT person.
 person->set_name( 'HITESH BHANUSHALI 05' ).
 person->set_age( 23 ).
 WRITE: / 'Name:', person->get_name( ),
        / 'Age:', person->get_age( ).
```

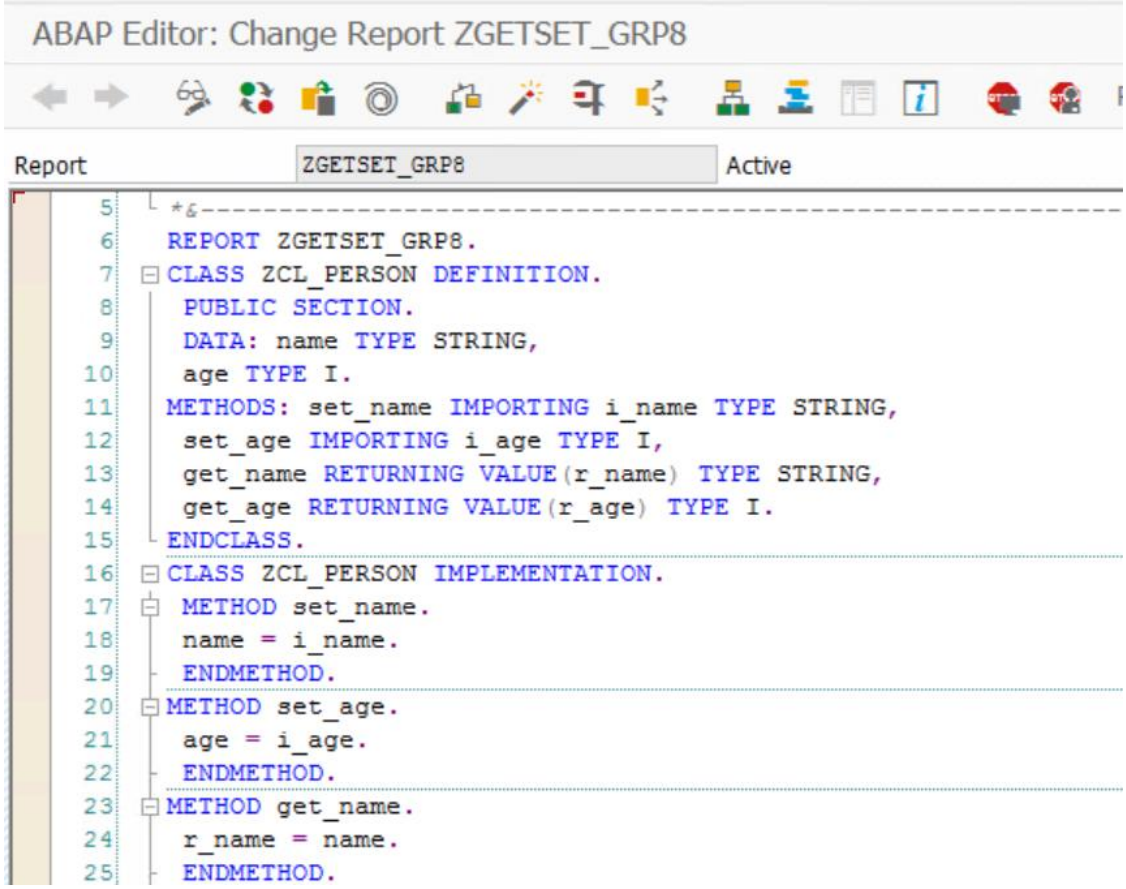
Steps to Perform

1. Log into the SAP System: Open the SAP GUI and log in to your SAP system.

2. Access ABAP Editor:

- Go to the transaction code SE38 or SE80.
- In SE38, type a program name (e.g., ZGETSET_GRP2) and click on Create.


ABAP Editor: Initial Screen
Debugging
With Variant
Variants
Program
ZGETSET_GRP8
Create
ABAP: Program Attributes ZGETSET_GRP8 Change
Title
Get and Set user input
Original language
EN
Created
FS4ABAP117
15.11.2024
Last Changed
FS4ABAP117
15.11.2024
Status
New (Revised)
Attributes
Type
Executable program
Status
Unclassified


ABAP Editor: Change Report ZGETSET_GRP8
Report
ZGETSET_GRP8
Active
*&---
REPORT ZGETSET_GRP8.
CLASS ZCL_PERSON DEFINITION.
PUBLIC SECTION.
DATA: name TYPE STRING,
age TYPE I.
METHODS: set_name IMPORTING i_name TYPE STRING,
set_age IMPORTING i_age TYPE I,
get_name RETURNING VALUE(r_name) TYPE STRING,
get_age RETURNING VALUE(r_age) TYPE I.
ENDCLASS.
CLASS ZCL_PERSON IMPLEMENTATION.
METHOD set_name.
name = i_name.
ENDMETHOD.
METHOD set_age.
age = i_age.
ENDMETHOD.
METHOD get_name.
r_name = name.
ENDMETHOD.

ABAP Editor: Change Report ZGETSET_GRP8

Report ZGETSET_GRP8 Active

```
METHOD set_name.
name = i_name.
ENDMETHOD.
METHOD set_age.
age = i_age.
ENDMETHOD.
METHOD get_name.
r_name = name.
ENDMETHOD.
METHOD get_age.
r_age = age.
ENDMETHOD.
ENDCLASS.
START-OF-SELECTION.
DATA: person TYPE REF TO ZCL_PERSON.
CREATE OBJECT person.
person->set_name( 'HITESH BHANUSHALI 05' ).
person->set_age( 23 ).
WRITE: / 'Name:', person->get_name( ),
 / 'Age:', person->get_age( ).
```

OUTPUT:

Get and Set user input

```
Get and Set user input

Name: HITESH BHANUSHALI 05
Age:          23
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No 7

Aim: Develop an ABAP program that performs a join between two tables and displays the results. Ensure that the join is based on a common key.

Theory: In ABAP, database joins enable data retrieval from multiple tables based on a common field. Joins are essential for efficiently combining related data in SAP applications, such as combining employee and department tables to retrieve an employee's department details. This is typically done using the INNER JOIN clause in SQL statements, specifying the common key and selecting fields from both tables. Joins in ABAP are used in reporting, data analysis, and business processes requiring information from multiple tables. They eliminate the need for multiple queries, reducing database load and improving performance. For example, in an employee report, joining employee and department tables on a shared field (e.g., dept_id) provides a complete view of each employee's details along with department information. ABAP's SQL interface makes it straightforward to implement joins, supporting efficient data retrieval and optimized reporting in SAP.

Tables:

Table-1: **ZEMP_ACCC**

Table-2**: ZDEPT_ACCC**

Code:

```
REPORT ZJOIN_GRP8.

Declare internal table and work area for the joined data

TYPES: BEGIN OF ty_empdept,

 emp_id TYPE ZEMP_ACCC-ID,

 emp_name TYPE ZEMP_ACCC-NAME,

 dept_name TYPE ZDEPT_ACCC-DEPT_NAME,

 END OF ty_empdept.

DATA: lt_empdept TYPE TABLE OF ty_empdept,

 ls_empdept TYPE ty_empdept.

Select data using INNER JOIN

SELECT a~ID AS emp_id, a~NAME AS emp_name, b~DEPT_NAME AS dept_name

 INTO CORRESPONDING FIELDS OF TABLE @lt_empdept

 FROM ZEMP_ACCC AS a

 INNER JOIN ZDEPT_ACCC AS b ON a~DEPTID = b~DEPTID.
```

Display the joined data

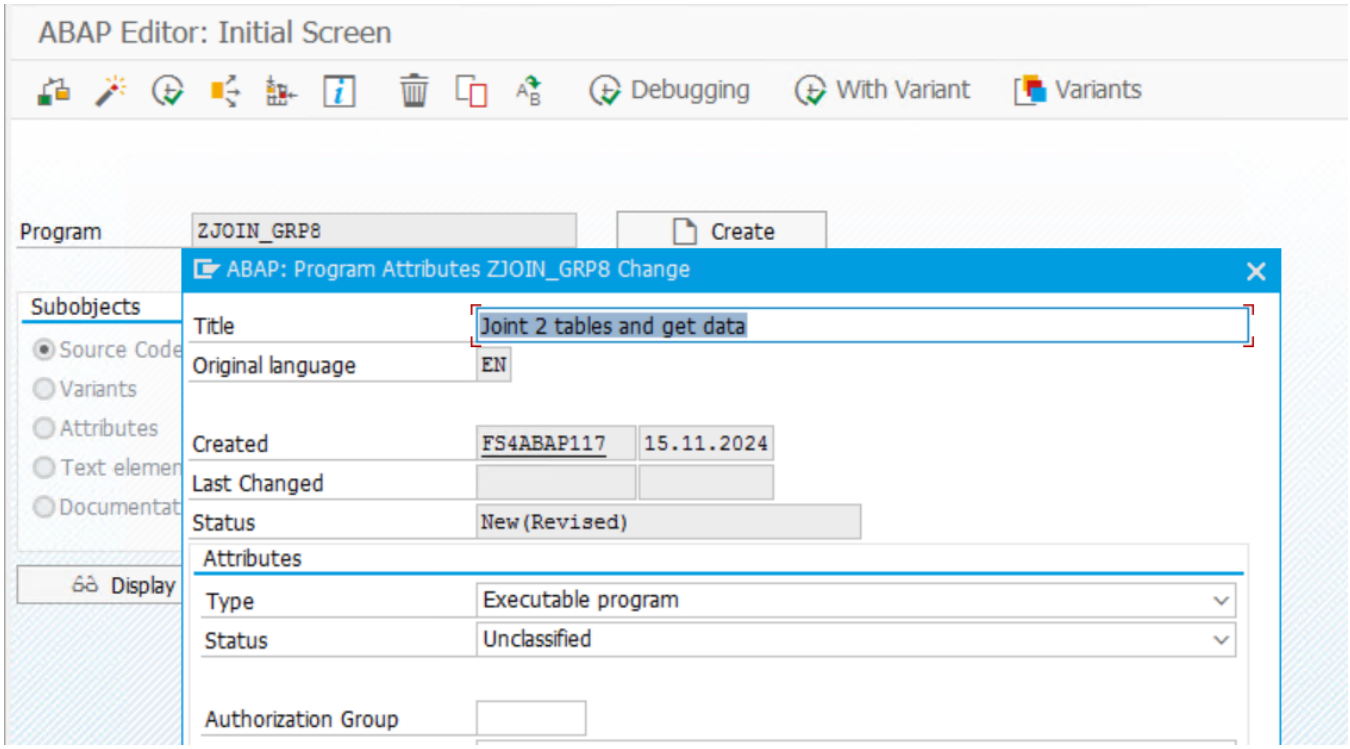
```
LOOP AT lt_empdept INTO ls_empdept.
 WRITE: / 'Employee ID:', ls_empdept-emp_id,
 / 'Employee Name:', ls_empdept-emp_name,
 / 'Department:', ls_empdept-dept_name.
ENDLOOP.
```

Output:

1. Log into the SAP System: Open the SAP GUI and log in to your SAP system.

2. Access ABAP Editor:

- Go to the transaction code SE38 or SE80.
- In SE38, type a program name (e.g., ZJOIN_GRP8) and click on Create.
- Write the code in the ABAP Editor



ABAP Editor: Change Report ZJOIN_GRP8

Report ZJOIN_GRP8 Active

```
*&
*&---------------------------------------------------------------------*
REPORT ZJOIN_GRP8.
**Declare internal table and work area for the joined data
TYPES: BEGIN OF ty_empdept,
           emp_id TYPE ZEMP_ACCC-ID,
           emp_name TYPE ZEMP_ACCC-NAME,
           dept_name TYPE ZDEPT_ACCC-DEPT_NAME,
       END OF ty_empdept.
DATA: lt_empdept TYPE TABLE OF ty_empdept,
      ls_empdept TYPE ty_empdept.
**Select data using INNER JOIN
SELECT a~ID AS emp_id, a~NAME AS emp_name, b~DEPT_NAME AS dept_name
  INTO CORRESPONDING FIELDS OF TABLE @lt_empdept
  FROM ZEMP_ACCC AS a
  INNER JOIN ZDEPT_ACCC AS b ON a~DEPTID = b~DEPTID.
LOOP AT lt_empdept INTO ls_empdept.
  WRITE: / 'Employee ID:', ls_empdept-emp_id,
         / 'Employee Name:', ls_empdept-emp_name,
         / 'Department:', ls_empdept-dept_name.
ENDLOOP.
```

3. Save and Check Syntax:

- Click Save.
- Click Check (or press Ctrl + F2) to ensure the code is free of errors.

4. Activate the Program:

- Click Activate (or press Ctrl + F3) to activate the program**.**

5. Run the Program:

- Click Execute (or press F8) to run the program

OUTPUT:

Joint 2 tables and get data

```
Joint 2 tables and get data

Employee ID:              1
Employee Name: Hitesh Bhanusha
Department: MSCIT
Employee ID:              2
Employee Name: Mohit Wadhwa
Department: MSCIT
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 8

Aim: Describe and execute a debugging session for an ABAP program that is not producing the expected output. Use breakpoints and watchpoints to identify the issue.

Theory:

Debugging is an essential skill in ABAP, allowing developers to identify and resolve issues within their programs. SAP's debugger provides powerful tools for examining variable values, inspecting the program's execution flow, and tracking data changes. Key debugging tools include breakpoints (which pause execution at specific points) and watchpoints (which pause execution when a specific condition is met). Debugging allows developers to step through code line-by-line, observe changes in real-time, and troubleshoot unexpected behavior. For example, in a payroll program, a developer can use the debugger to check if variables are correctly updated through calculations and logic flows. By carefully examining the program's execution, developers can identify errors, validate assumptions, and ensure that programs meet business requirements. SAP's debugging tools make it easier to troubleshoot complex ABAP applications and deliver reliable, high-quality code.

Steps:

Step-1. Set Breakpoints

- Open the program in SE38 or SE80.
- Set breakpoints by clicking the left margin next to the line number where you suspect the issue.

Step-2. Activate Debugging Mode

- Execute the program using F8.
- The program will stop at the breakpoint and enter debugging mode.

Step-3. Inspect Variables

- In the Debugger, check the Variables tab for the values of relevant variables.
- Hover over variables or manually inspect them.

Step-4. Step Through Code

- Use F5 (Step Into), F6 (Step Over), and F7 (Return) to step through the program line by line.

Step-5. Set Watchpoints (Optional)

- Set a Watchpoint on a variable by right-clicking it and selecting "Create Watchpoint" to monitor specific
conditions.

Step-6. Analyze the Flow

- Check if loops, conditions, and calculations are working as expected.

Step-7. Fix the Issue

- Modify code or change variable values in the debugger to test fixes.
- Once identified, fix the logic in the program.

Step-8. Run Again

- After making changes, run the program again to ensure the issue is resolved.

Step-9. End Debugging

- Click Stop Debugging to exit the debugger when done.

ABAP Editor: Change Report ZJOIN_GRP8

Report ZJOIN_GRP8 Active

```
REPORT ZJOIN_GRP8.

**Declare internal table and work area for the joined data
TYPES: BEGIN OF ty_empdept,
           emp_id TYPE ZEMP_ACCC-ID,
           emp_name TYPE ZEMP_ACCC-NAME,
           dept_name TYPE ZDEPT_ACCC-DEPT_NAME,
       END OF ty_empdept.
DATA: lt_empdept TYPE TABLE OF ty_empdept,
      ls_empdept TYPE ty_empdept.
**Select data using INNER JOIN
SELECT a~ID AS emp_id, a~NAME AS emp_name, b~DEPT_NAME AS dept_name
  INTO CORRESPONDING FIELDS OF TABLE @lt_empdept
  FROM ZEMP_ACCC AS a
  INNER JOIN ZDEPT_ACCC AS b ON a~DEPTID = b~DEPTID.
LOOP AT lt_empdept INTO ls_empdept.
  WRITE: / 'Employee ID:', ls_empdept-emp_id,
         / 'Employee Name:', ls_empdept-emp_name,
         / 'Department:', ls_empdept-dept_name.
ENDLOOP.
```

ABAP Debugger(1) (Exclusive) (s4hana2022_A4H_52)

Step Size | Watchpoint | Layout | Configure Debugger Layer

ZJOIN_GRP8 / ZJOIN_GRP8 / 22 SY-SUBRC 0

EVENT / START-OF-SELECTION SY-TABIX 1

Desktop 1 | Desktop 2 | Desktop 3 | Standard | Structures | Tables | Objects | Detail | Data Explorer | Break./Watchpoints | Diff | Script

```
          dept_name TYPE ZDEPT_ACCC-DEPT_NAME,
       END OF ty_empdept.
DATA: lt_empdept TYPE TABLE OF ty_empdept,
      ls_empdept TYPE ty_empdept.
**Select data using INNER JOIN
SELECT a~ID AS emp_id, a~NAME AS emp_name, b~DEPT_NAME AS dept
  INTO CORRESPONDING FIELDS OF TABLE @lt_empdept
  FROM ZEMP_ACCC AS a
  INNER JOIN ZDEPT_ACCC AS b ON a~DEPTID = b~DEPTID.
LOOP AT lt_empdept INTO ls_empdept.
  WRITE: / 'Employee ID:', ls_empdept-emp_id,
         / 'Employee Name:', ls_empdept-emp_name,
         / 'Department:', ls_empdept-dept_name.
ENDLOOP.
```

ABAP and Screen Stack

St...	Sta...	S..	Event Type	Event	Program	Na...	Incl
	2		EVENT	START-OF-SELECTION	ZJOIN_GRP8		ZJOI
	1		PAI SCREEN	1000	SAPMSSY0		

Variables 1 | Variables 2 | Locals | Globals | Auto | Memory Analysis

S...	Variable	V...	Val.	C...	Hexadecimal Value

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

Practical No. 9

Aim: Explain the key differences between SAP ECC and S/4HANA in terms of data model and performance. Include examples of how ABAP programming might need to adapt.

Theory:

The key differences between SAP ECC and S/4HANA are as follows:

1. Underlying Database

- SAP ECC: Runs on traditional disk-based relational databases like Oracle, IBM DB2, or Microsoft SQL Server. Data retrieval can be slower, especially with large datasets.
- SAP S/4HANA: Built on the SAP HANA in-memory database, which stores data in RAM, enabling much faster data processing and real-time analytics.

2. Data Model

- SAP ECC: Uses a complex data model with numerous aggregate and index tables to improve performance, often leading to redundancy and inefficient data management.
- SAP S/4HANA: Simplifies the data model by removing redundant tables and aggregates, reducing the data footprint and enabling faster data access and real-time analytics.

3. Performance

- SAP ECC: Performance depends on the underlying database, requiring complex optimization for large datasets, leading to slower transaction times and delays in reporting.
- SAP S/4HANA: Takes full advantage of the HANA in-memory platform for real-time data processing, improving transaction times, reporting speed, and overall business agility.

4. User Interface (UI)

- SAP ECC: Uses the traditional SAP GUI, which is functional but complex and less intuitive for endusers.
- SAP S/4HANA: Features the modern SAP Fiori UI, which is role-based, responsive, and accessible across multiple devices (desktop, tablet, mobile), offering a more user-friendly experience.

5. Functionality

- SAP ECC: Provides core ERP functionalities but often requires additional components or customizations for advanced features like predictive analytics.
- SAP S/4HANA: Includes enhanced capabilities out of the box, such as predictive analytics, machine learning, and embedded analytics, allowing businesses to gain deeper insights and make data-driven decisions.

6. Customization and Extensibility

- SAP ECC: Customization is typically done through ABAP enhancements and modifications, which can be time-consuming and complex.
- SAP S/4HANA: Promotes a more flexible approach to customization using the SAP Cloud Platform and extensibility frameworks, allowing businesses to easily tailor the system to their needs.

7. Development Paradigm

- SAP ECC: Customization and development primarily rely on traditional ABAP programming techniques, such as classical reports and database accesses.
- SAP S/4HANA: Introduces modern development tools and paradigms, such as Core Data Services (CDS) views, ABAP Managed Database Procedures (AMDP), OData services, and Fiori integration for frontend/backend communication.

8. Migration and Refactoring

- SAP ECC: Migrating to ECC often requires manual modifications, complex integrations, and adjustments to ensure compatibility with other systems.
- SAP S/4HANA: The transition to S/4HANA requires refactoring ABAP code to accommodate the simplified data model, more efficient querying, and the use of new tools and methodologies like CDS views and AMDP.

9. Integration with Cloud and Modern Tools

- SAP ECC: Integrating with cloud-based applications and modern tools requires additional setup, custom code, and middleware.
- SAP S/4HANA: Built with native cloud integration capabilities, enabling seamless connections to cloud-based applications and services, facilitating a more flexible IT ecosystem.