SAP ABAP Project Report

Project Title: Employee Leave Management System
Submitted by: ***Karthik S***
Under the guidance of: ***Mr. Rajendra***

A) Executive Summary

This report outlines the design, implementation, and testing of the Employee Leave Management System (ELMS) developed using SAP ABAP.
The system automates the management of employee leave records, integrating multiple ABAP components such as the Data Dictionary, Function Modules, Table Maintenance Generator (TMG), Reports, and Module Pool Programming.

This project demonstrates a full-cycle SAP ABAP implementation of employee leave management integrating all major technical components.

Introduction to SAP ABAP

SAP ABAP (Advanced Business Application Programming) is the core language used to develop applications within the SAP ecosystem.
ABAP enables customization of standard SAP modules and supports the creation of reports, user interfaces, and database management processes.

B) Employee Leave Management System

This SAP ABAP mini project demonstrates a complete Employee Leave Management System implemented using core ABAP concepts like Data Dictionary, Function Modules, Reports, and Module Pool Programming.
The project focuses on maintaining employee leave records, automating leave ID generation, and enabling CRUD operations with reporting functionality.

C)Objective

The objective of this project is to design and develop an Employee Leave Management System in SAP ABAP that allows users to:
1. Create and maintain leave records.
2. Automate leave ID generation using Function Modules.
3. Display employee leave data using ALV reports.
4. Enable table maintenance through SM30 (Table Maintenance Generator).
5. Validate data entries and provide complete CRUD functionality.

System Overview

The Employee Leave Management System is designed using a three-layer architecture:
1. Presentation Layer – Handles user inputs through reports and screens.
2. Application Layer – Manages business logic via Function Modules.
3. Database Layer – Stores leave details using a custom transparent table in the Data Dictionary.

D) Modules Implemented

- 1. Data Dictionary Objects (Table, Domains, Data Elements)
- 2. Technical Settings Configuration
- 3. Table Maintenance Generator (TMG)
- 4. Function Module: Z_FM_ELM_CREATE
- 5. Include Program: ZELM_INCL_DB
- 6. Report Program: ZELM_CREATE_UI
- 7. Module Pool Program: ZELM_MGNT
- 8. ALV Report: ZELM_RPT
- 9. Data Browser Validation (SE16N)
- 10. Testing and Output Verification

E) Implementation Details

Each module in this system serves a specific function. The Data Dictionary defines custom objects,
the Function Module automates unique ID generation, and Reports display output in structured ALV format.
The Module Pool handles user interactions while TMG provides a simple interface for maintaining entries.

- ### Function Module - Z_FM_ELM_CREATE

This Function Module automates the process of inserting records into the ZELM_LEAVE table.
It ensures data validation and prevents duplicate Leave IDs by generating new unique identifiers.

- ### Reports - ZELM_CREATE_UI and ZELM_RPT

The ZELM_CREATE_UI report allows users to input employee leave details, while ZELM_RPT retrieves and displays leave records using ALV grids.
Both reports use modularized code from ZELM_INCL_DB for better maintainability.

- **Module Pool Program - ZELM_MGNT**

This program implements PBO (Process Before Output) and PAI (Process After Input) logic to create an interactive screen for leave management.
It handles data entry, validation, and user navigation efficiently.

- **Table Maintenance Generator (TMG)**

TMG provides a simple SM30-based interface for data maintenance.
Administrators can add, modify, or delete leave records through the automatically generated maintenance screens.

- **Testing and Validation**

Testing was performed using transactions SE11, SE16N, and SM30.
Sample data entries were created, validated, and displayed successfully.
Functionality such as record creation, modification, and deletion was confirmed through TMG.

- **Output Explanation**

The project outputs include screenshots from SAP GUI demonstrating each module's functionality —
from table creation to data maintenance and ALV report display.
All test cases produced expected results confirming system reliability.

- **Results**

The Employee Leave Management System achieved all objectives.
Data integrity was maintained, and business logic was accurately implemented through ABAP coding practices.

F) Tools and Environment

The Employee Leave Management System was developed and tested in the SAP ABAP environment using the following tools and system configurations:

1. Software Environment

- **SAP GUI Version:** SAP Logon 7.80 (or your current version)
- **SAP Application Server:** SAP ECC 6.0 / SAP S/4HANA 1909
- **ABAP Editor Transactions Used:** SE11, SE37, SE38, SE80, SE16N, SM30
- **Database:** SAP HANA (Integrated)
- **Operating System:** Windows 10 / 11 (64-bit)
- **Programming Language:** ABAP (Advanced Business Application Programming)

2. Hardware Requirements

- **Processor:** Intel Core i5 or higher
- **RAM:** Minimum 8 GB (Recommended 16 GB for SAP local systems)
- **Storage:** At least 20 GB free space for SAP environment

3. Development and Execution Environment

- **Development Server:** SAP ABAP Workbench (Object Navigator – SE80)
- **Testing Tools:** Data Browser (SE16N), Table Maintenance Generator (SM30)
- **Version Control:** TR Transport Requests within SAP
- **Output Tools:** ALV Grid Display for reports

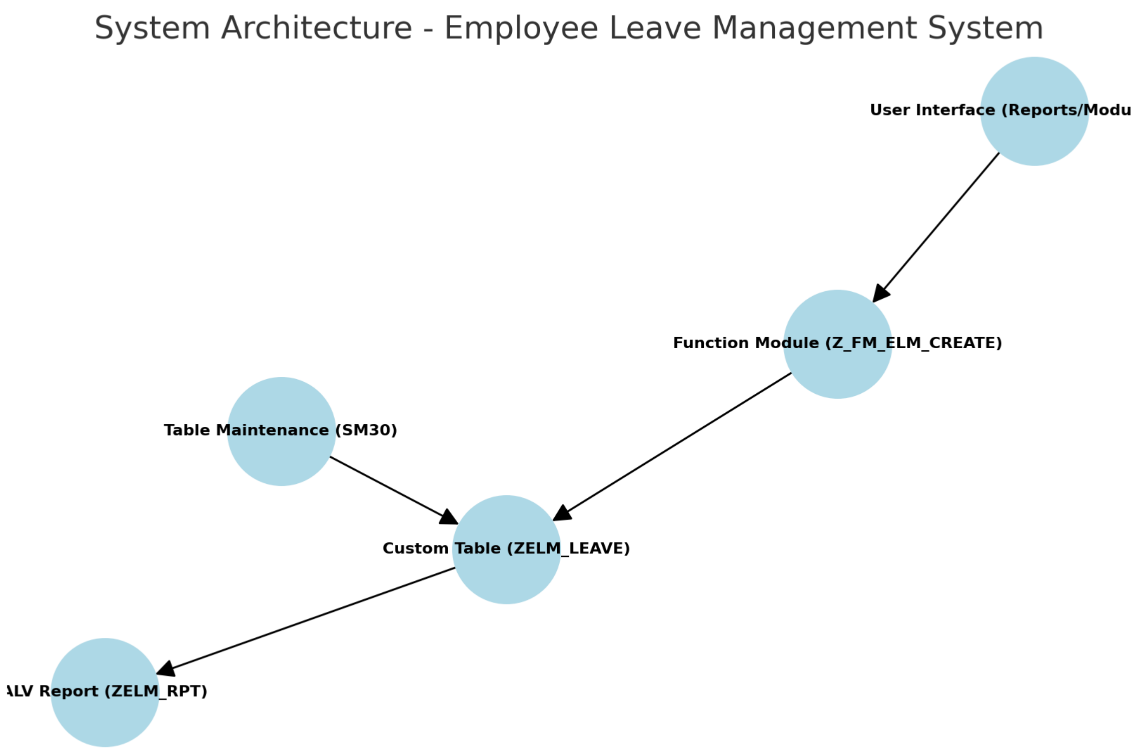
4. User Roles

- **Developer:** Responsible for creating ABAP programs and Function Modules
- **Administrator:** Handles TMG and authorization access via SM30
- **End User:** Executes reports and views leave data in ALV format

G) Project Flow

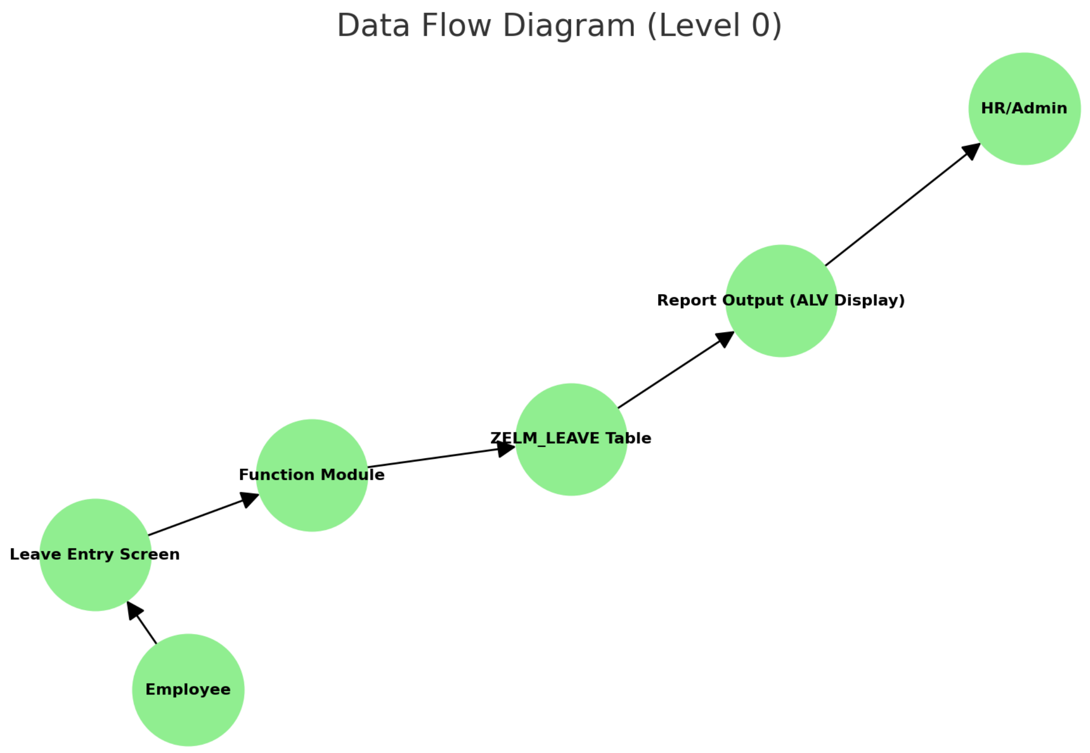
1. User enters leave details in Report ZELM_CREATE_UI.
2. Data is passed to Function Module Z_FM_ELM_CREATE which auto-generates a unique Leave ID.
3. The record is inserted into custom table ZELM_LEAVE.
4. Admins can maintain records via Table Maintenance Generator (SM30).
5. Employees and HR can view data via ALV Report ZELM_RPT.
6. Module Pool ZELM_MGNT handles screen interactions and validations.

- **System Architecture**


System Architecture - Employee Leave Management System
User Interface (Reports/Modu
Function Module (Z_FM_ELM_CREATE)
Table Maintenance (SM30)
Custom Table (ZELM_LEAVE)
ALV Report (ZELM_RPT)

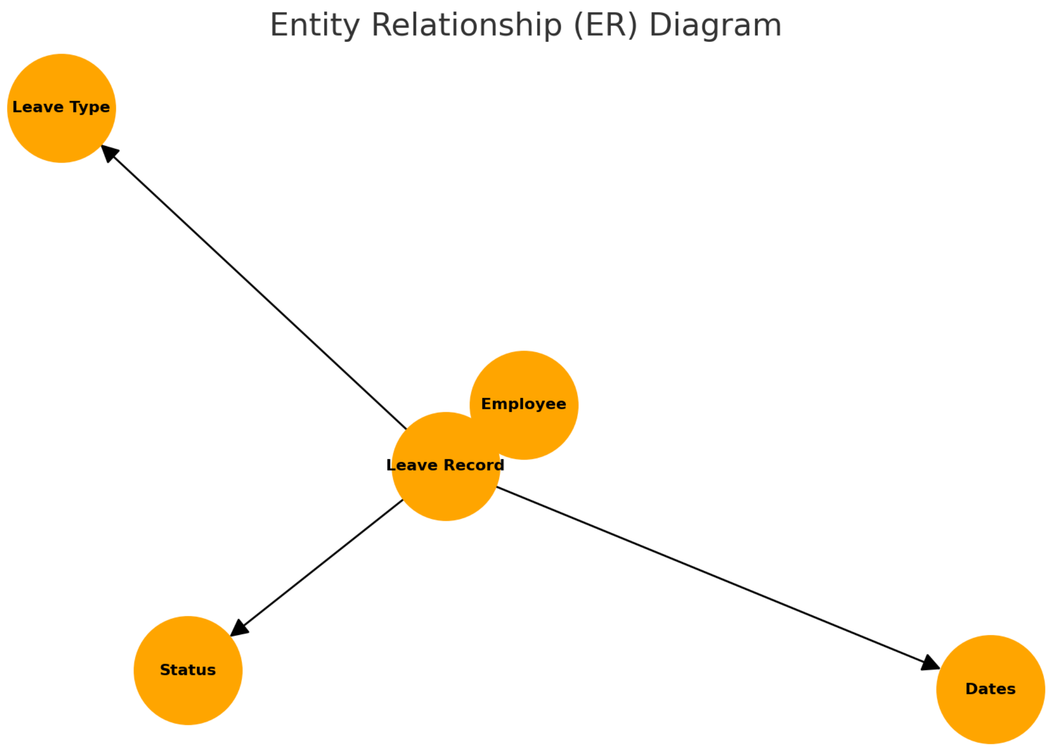
This diagram shows the three-tier architecture: UI, Function Module, and Database integration.

- **Data Flow Diagram (Level 0)**



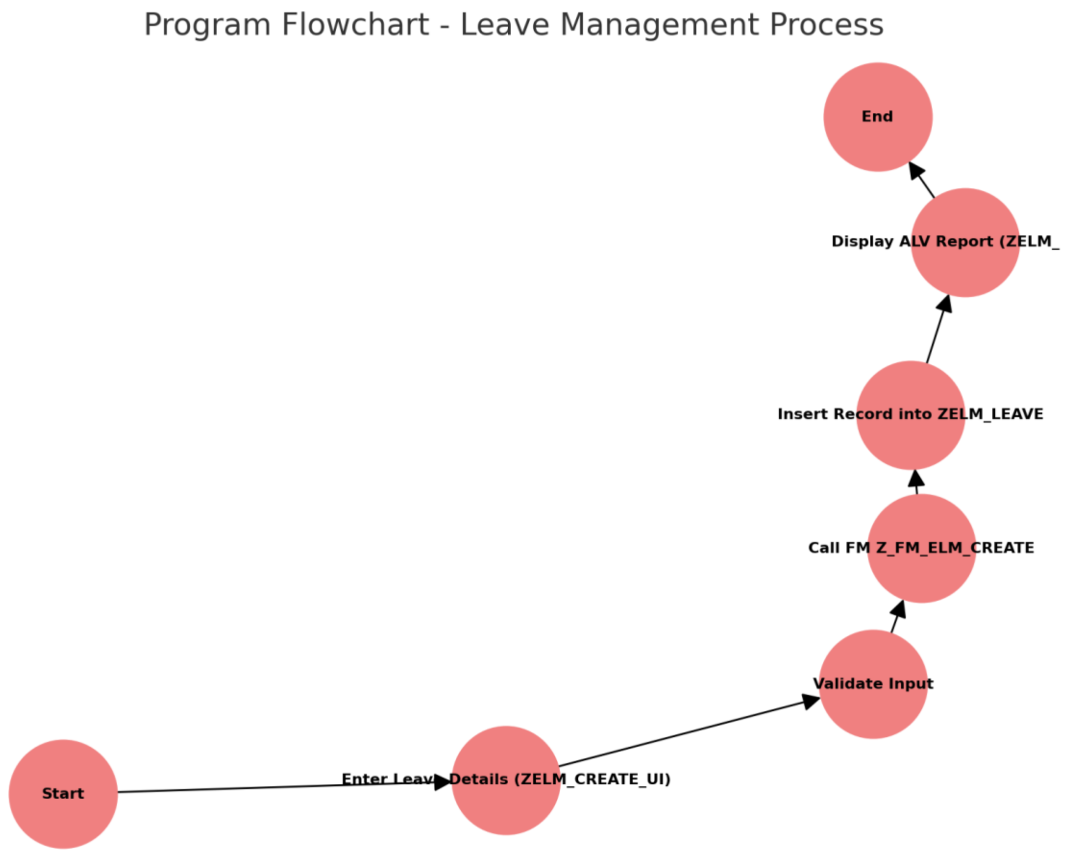
Depicts how employee data moves through modules from entry to reporting.

- **Entity Relationship Diagram**



Illustrates relationships between Employee, Leave Records, Types, and Status.

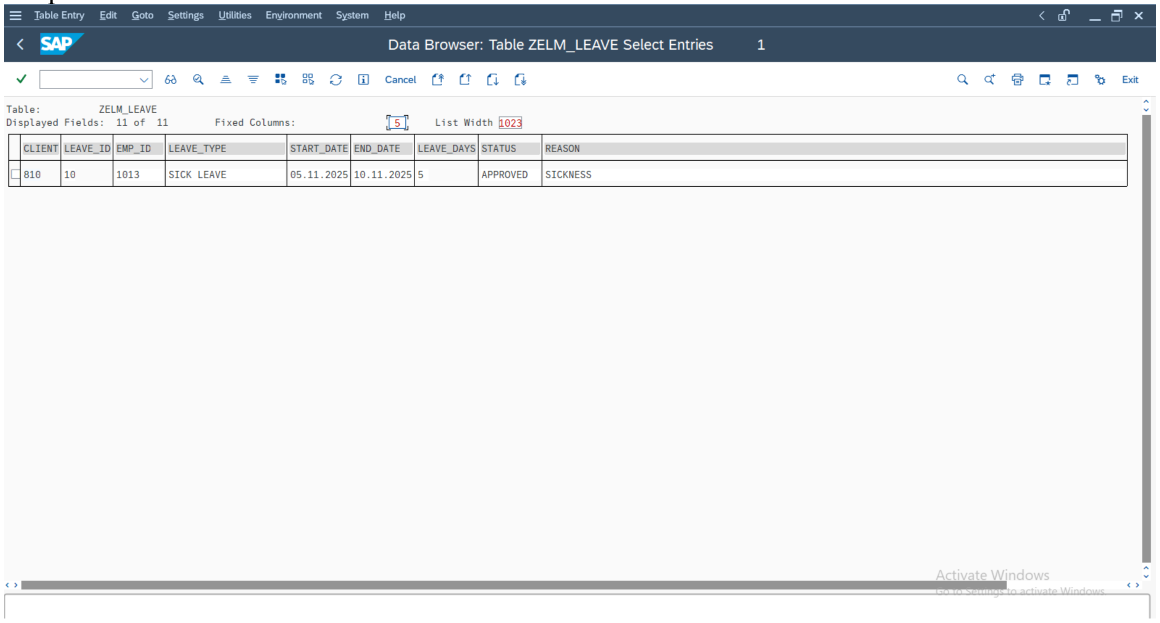
- **Program Flowchart**


Program Flowchart - Leave Management Process
Start
Enter Leave Details (ZELM_CREATE_UI)
Validate Input
Call FM Z_FM_ELM_CREATE
Insert Record into ZELM_LEAVE
Display ALV Report (ZELM_
End

Flowchart visualizing main ABAP logic from user input to report generation.

H) Output Screenshots

Output 1

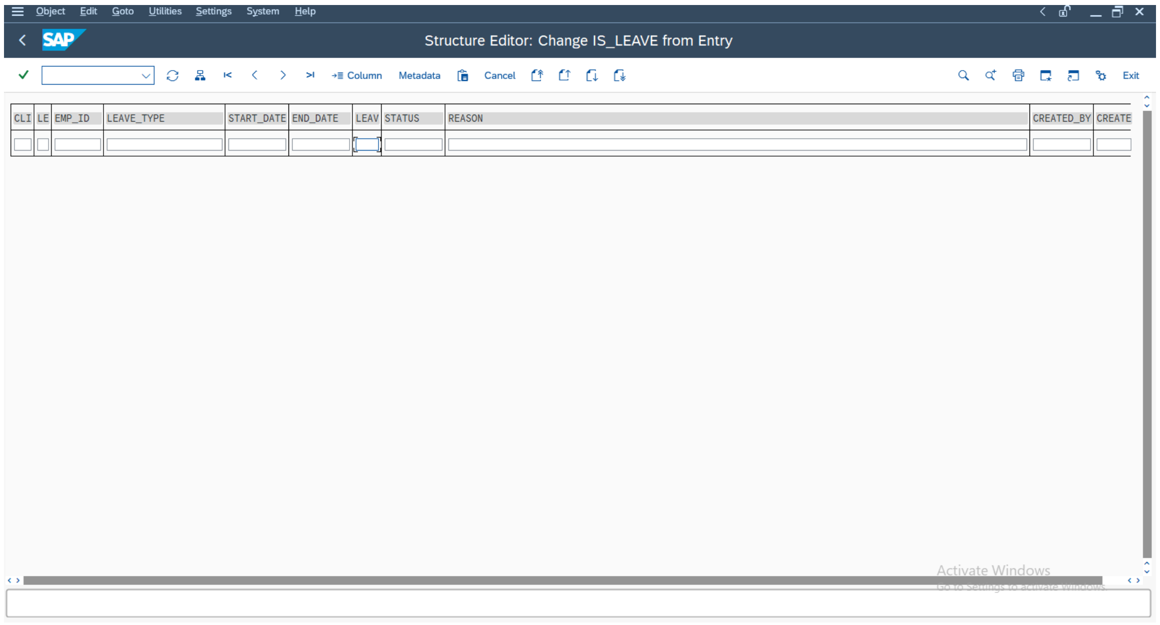


Data Browser: Table ZELM_LEAVE Select Entries 1

```
Table:          ZELM_LEAVE
Displayed Fields:  11 of  11        Fixed Columns:          5     List Width 1023
```

CLIENT	LEAVE_ID	EMP_ID	LEAVE_TYPE	START_DATE	END_DATE	LEAVE_DAYS	STATUS	REASON
810	10	1013	SICK LEAVE	05.11.2025	10.11.2025	5	APPROVED	SICKNESS

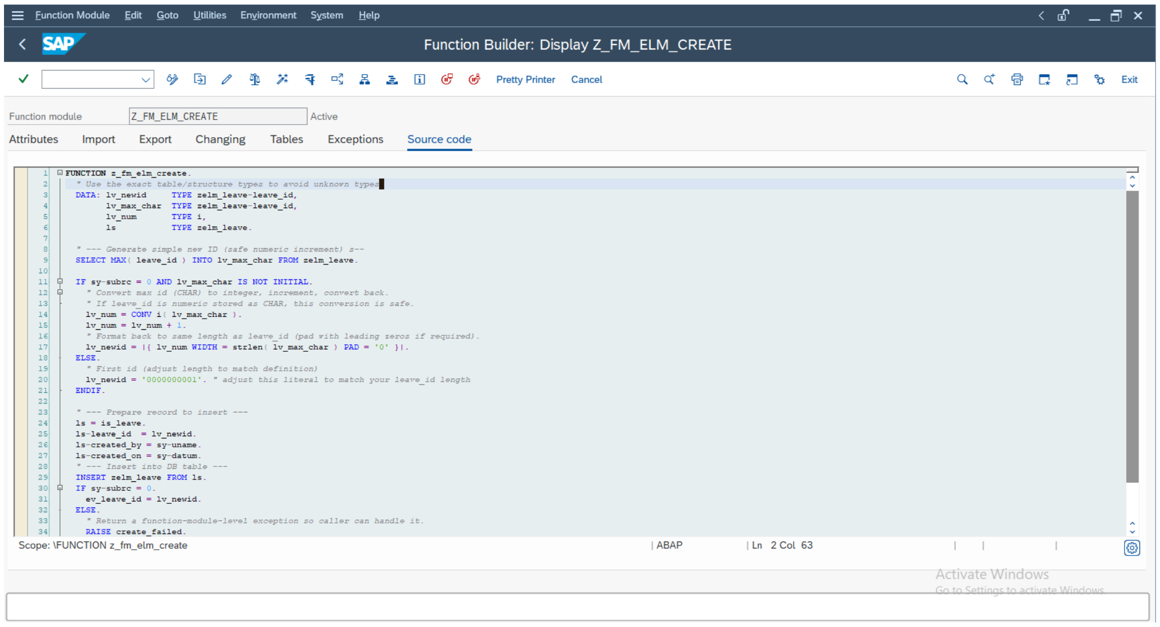
Output 2



Structure Editor: Change IS_LEAVE from Entry

CLI	LE	EMP_ID	LEAVE_TYPE	START_DATE	END_DATE	LEAV	STATUS	REASON	CREATED_BY	CREATE

Output 3

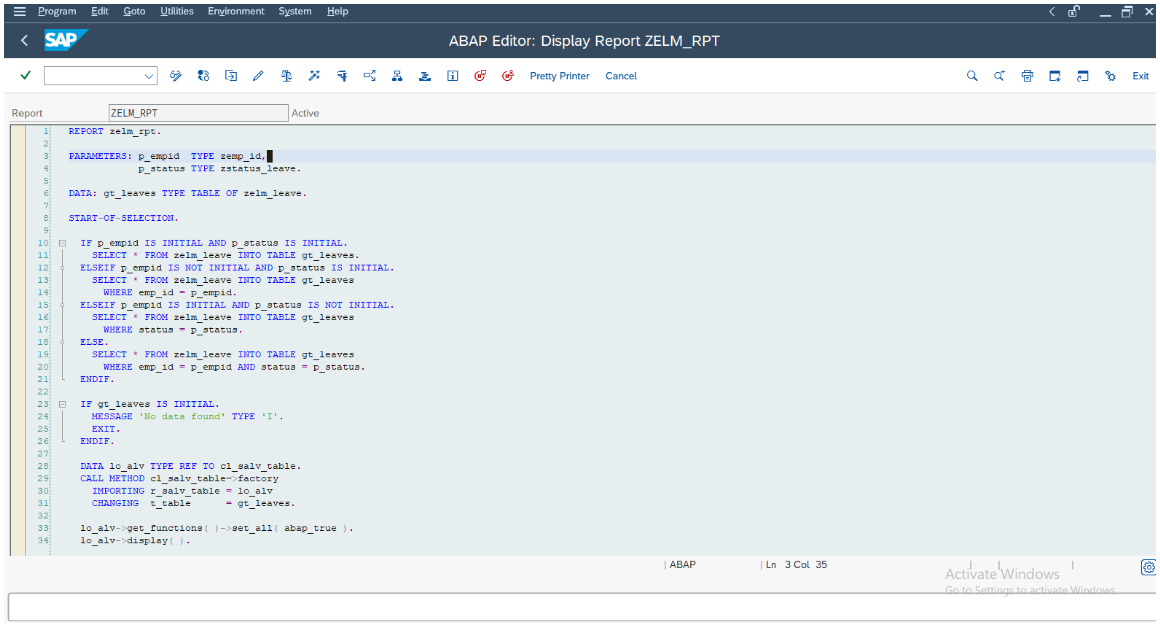

Function Module Edit Goto Utilities Environment System Help
Function Builder: Display Z_FM_ELM_CREATE
Pretty Printer Cancel Exit
Attributes Import Export Changing Tables Exceptions Source code
| ABAP | Ln 2 Col 63

Function module Z_FM_ELM_CREATE Active

```
FUNCTION z_fm_elm_create.
  " Use the exact table/structure types to avoid unknown types
  DATA: lv_newid     TYPE zelm_leave-leave_id,
        lv_max_char  TYPE zelm_leave-leave_id,
        lv_num       TYPE i,
        ls           TYPE zelm_leave.

  " --- Generate simple new ID (safe numeric increment) s--
  SELECT MAX( leave_id ) INTO lv_max_char FROM zelm_leave.

  IF sy-subrc = 0 AND lv_max_char IS NOT INITIAL.
    " Convert max id (CHAR) to integer, increment, convert back.
    " If leave_id is numeric stored as CHAR, this conversion is safe.
    lv_num = CONV i( lv_max_char ).
    lv_num = lv_num + 1.
    " Format back to same length as leave_id (pad with leading zeros if required).
    lv_newid = |{ lv_num WIDTH = strlen( lv_max_char ) PAD = '0' }|.
  ELSE.
    " First id (adjust length to match definition)
    lv_newid = '0000000001'. " adjust this literal to match your leave_id length
  ENDIF.

  " --- Prepare record to insert ---
  ls = is_leave.
  ls-leave_id  = lv_newid.
  ls-created_by = sy-uname.
  ls-created_on = sy-datum.
  " --- Insert into DB table ---
  INSERT zelm_leave FROM ls.
  IF sy-subrc = 0.
    ev_leave_id = lv_newid.
  ELSE.
    " Return a function-module-level exception so caller can handle it.
    RAISE create_failed.
```

Scope: \FUNCTION z_fm_elm_create

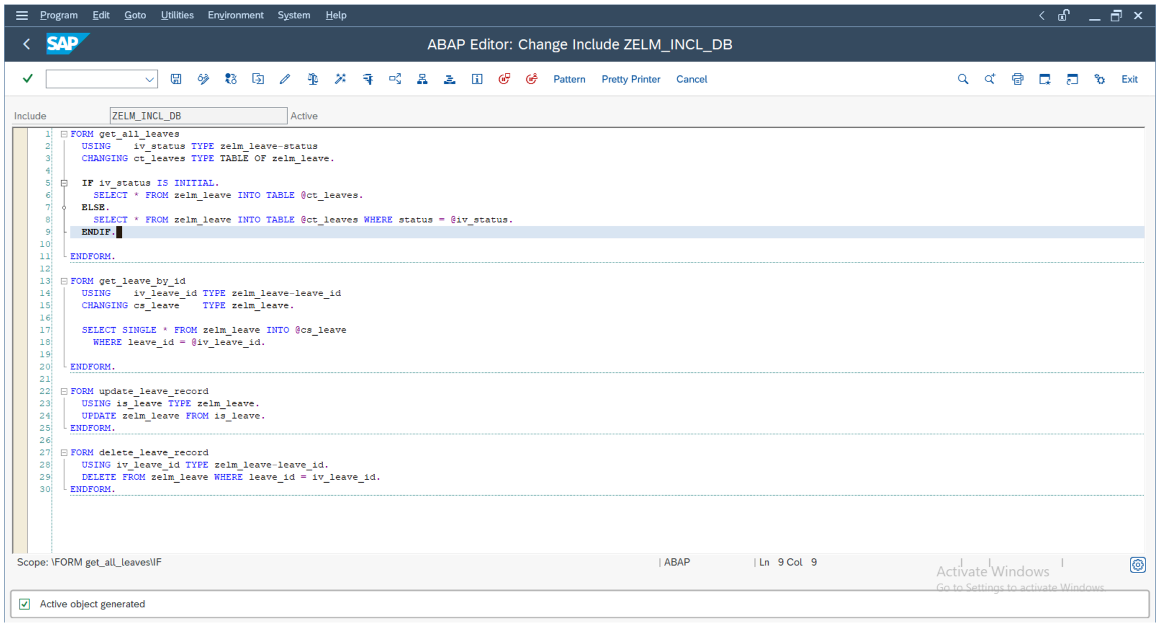
Output 4


Program Edit Goto Utilities Environment System Help
ABAP Editor: Display Report ZELM_RPT
Pretty Printer Cancel Exit
| ABAP | Ln 3 Col 35

Report ZELM_RPT Active

```
REPORT zelm_rpt.

PARAMETERS: p_empid  TYPE zemp_id,
            p_status TYPE zstatus_leave.

DATA: gt_leaves TYPE TABLE OF zelm_leave.

START-OF-SELECTION.

  IF p_empid IS INITIAL AND p_status IS INITIAL.
    SELECT * FROM zelm_leave INTO TABLE gt_leaves.
  ELSEIF p_empid IS NOT INITIAL AND p_status IS INITIAL.
    SELECT * FROM zelm_leave INTO TABLE gt_leaves
      WHERE emp_id = p_empid.
  ELSEIF p_empid IS INITIAL AND p_status IS NOT INITIAL.
    SELECT * FROM zelm_leave INTO TABLE gt_leaves
      WHERE status = p_status.
  ELSE.
    SELECT * FROM zelm_leave INTO TABLE gt_leaves
      WHERE emp_id = p_empid AND status = p_status.
  ENDIF.

  IF gt_leaves IS INITIAL.
    MESSAGE 'No data found' TYPE 'I'.
    EXIT.
  ENDIF.

  DATA lo_alv TYPE REF TO cl_salv_table.
  CALL METHOD cl_salv_table=>factory
    IMPORTING r_salv_table = lo_alv
    CHANGING  t_table      = gt_leaves.

  lo_alv->get_functions( )->set_all( abap_true ).
  lo_alv->display( ).
```

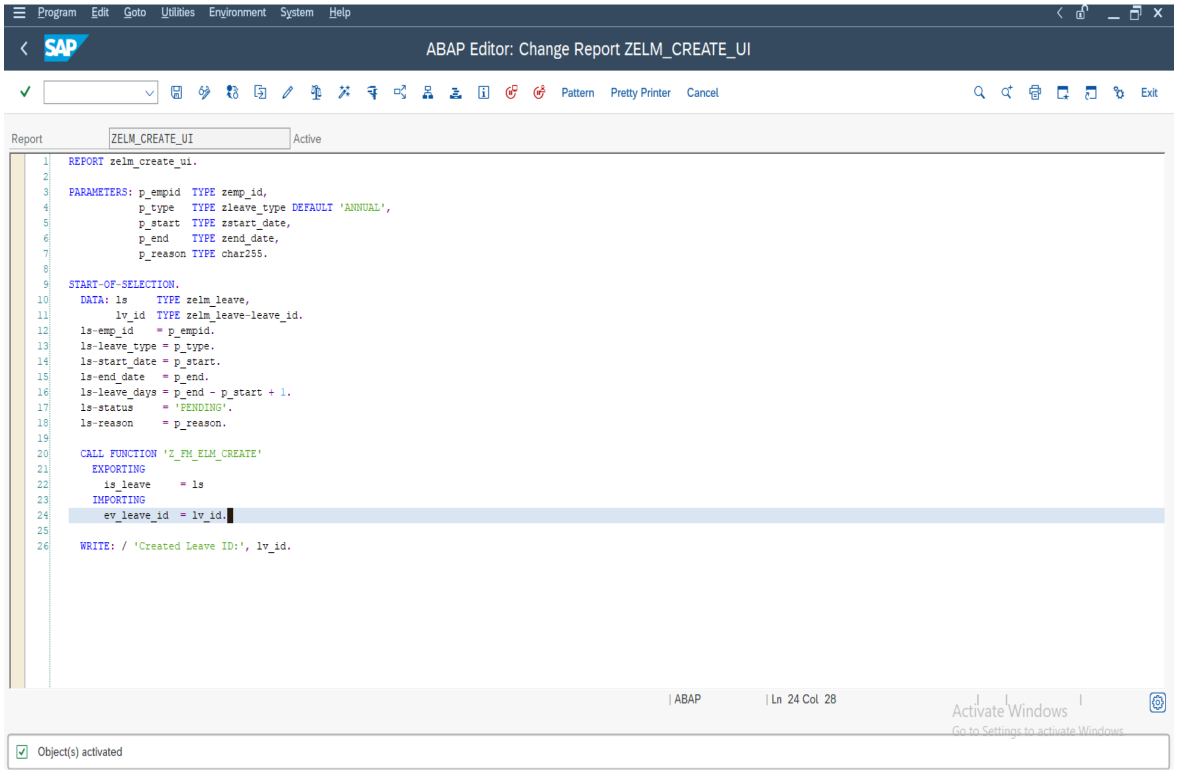
Output 5


Program Edit Goto Utilities Environment System Help
ABAP Editor: Change Include ZELM_INCL_DB
Pattern Pretty Printer Cancel Exit
Include ZELM_INCL_DB Active
Scope: \FORM get_all_leaves\IF
ABAP Ln 9 Col 9

```
FORM get_all_leaves
  USING    iv_status TYPE zelm_leave-status
  CHANGING ct_leaves TYPE TABLE OF zelm_leave.

  IF iv_status IS INITIAL.
    SELECT * FROM zelm_leave INTO TABLE @ct_leaves.
  ELSE.
    SELECT * FROM zelm_leave INTO TABLE @ct_leaves WHERE status = @iv_status.
  ENDIF.

ENDFORM.

FORM get_leave_by_id
  USING    iv_leave_id TYPE zelm_leave-leave_id
  CHANGING cs_leave    TYPE zelm_leave.

  SELECT SINGLE * FROM zelm_leave INTO @cs_leave
    WHERE leave_id = @iv_leave_id.

ENDFORM.

FORM update_leave_record
  USING is_leave TYPE zelm_leave.
  UPDATE zelm_leave FROM is_leave.
ENDFORM.

FORM delete_leave_record
  USING iv_leave_id TYPE zelm_leave-leave_id.
  DELETE FROM zelm_leave WHERE leave_id = iv_leave_id.
ENDFORM.
```

Active object generated

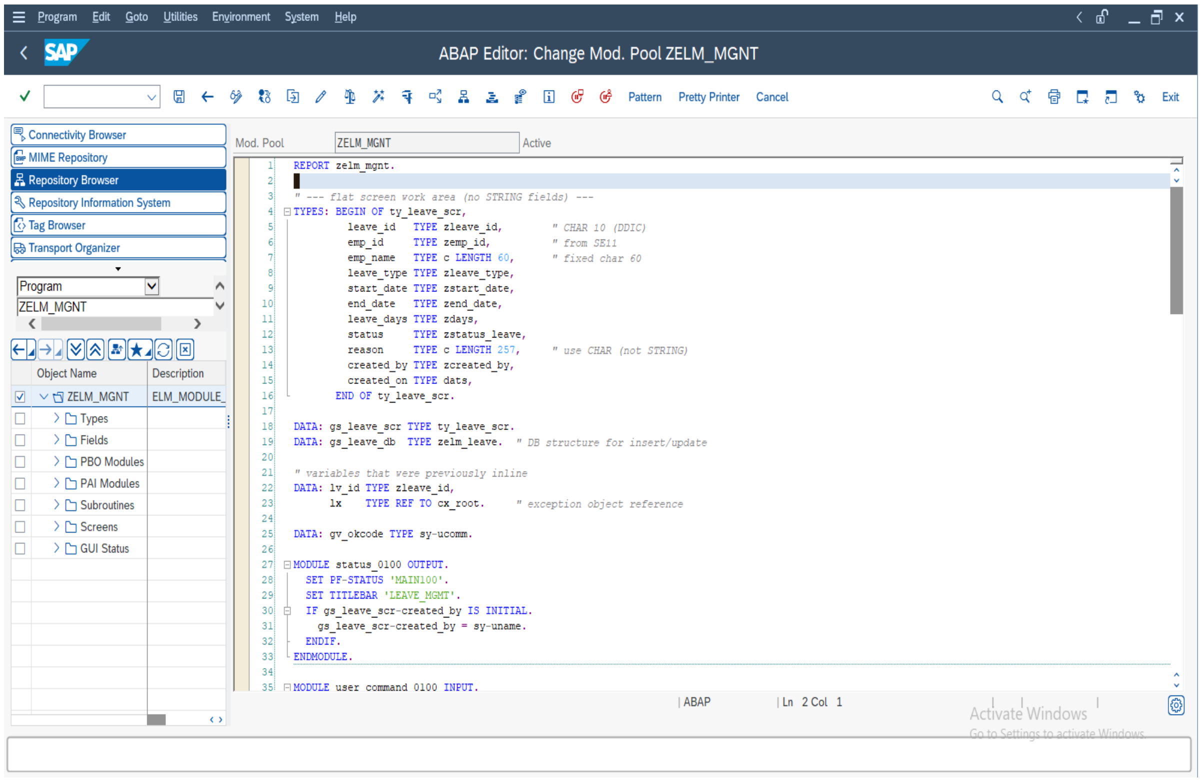
Output 6


Program Edit Goto Utilities Environment System Help
ABAP Editor: Change Report ZELM_CREATE_UI
Pattern Pretty Printer Cancel Exit
Report ZELM_CREATE_UI Active
ABAP Ln 24 Col 28

```
REPORT zelm_create_ui.

PARAMETERS: p_empid  TYPE zemp_id,
            p_type   TYPE zleave_type DEFAULT 'ANNUAL',
            p_start  TYPE zstart_date,
            p_end    TYPE zend_date,
            p_reason TYPE char255.

START-OF-SELECTION.
  DATA: ls      TYPE zelm_leave,
        lv_id  TYPE zelm_leave-leave_id.
  ls-emp_id     = p_empid.
  ls-leave_type = p_type.
  ls-start_date = p_start.
  ls-end_date   = p_end.
  ls-leave_days = p_end - p_start + 1.
  ls-status     = 'PENDING'.
  ls-reason     = p_reason.

  CALL FUNCTION 'Z_FM_ELM_CREATE'
    EXPORTING
      is_leave     = ls
    IMPORTING
      ev_leave_id  = lv_id.

  WRITE: / 'Created Leave ID:', lv_id.
```

Object(s) activated

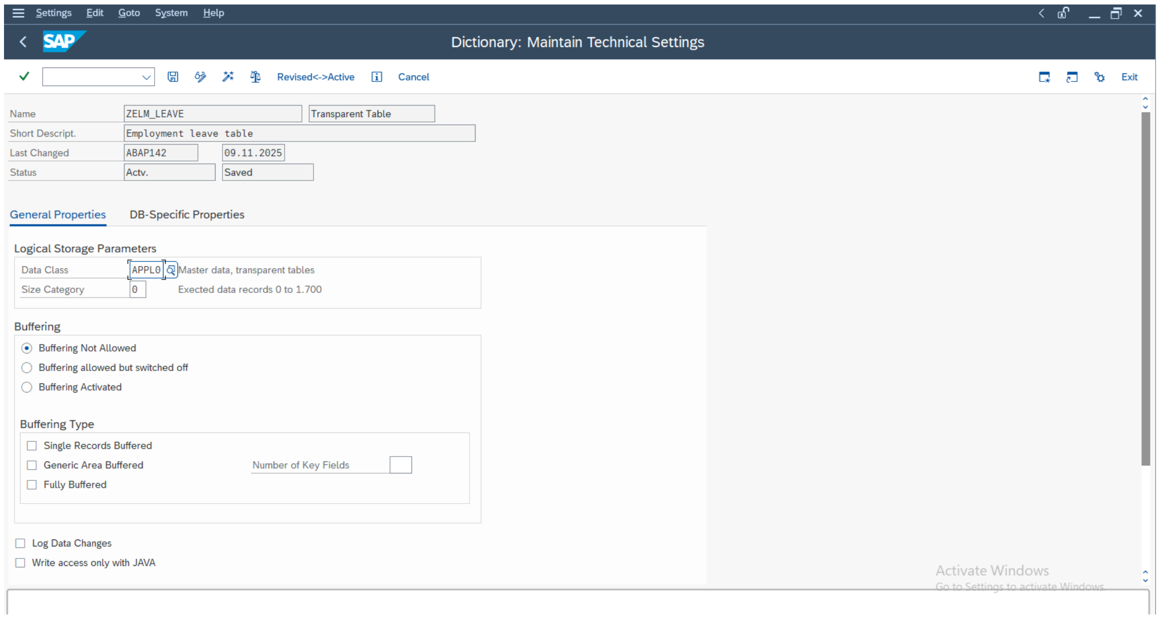
Output 7



Mod. Pool ZELM_MGNT Active

```
REPORT zelm_mgnt.

" --- flat screen work area (no STRING fields) ---
TYPES: BEGIN OF ty_leave_scr,
         leave_id   TYPE zleave_id,          " CHAR 10 (DDIC)
         emp_id     TYPE zemp_id,            " from SE11
         emp_name   TYPE c LENGTH 60,        " fixed char 60
         leave_type TYPE zleave_type,
         start_date TYPE zstart_date,
         end_date   TYPE zend_date,
         leave_days TYPE zdays,
         status     TYPE zstatus_leave,
         reason     TYPE c LENGTH 257,       " use CHAR (not STRING)
         created_by TYPE zcreated_by,
         created_on TYPE dats,
       END OF ty_leave_scr.

DATA: gs_leave_scr TYPE ty_leave_scr.
DATA: gs_leave_db  TYPE zelm_leave.  " DB structure for insert/update

" variables that were previously inline
DATA: lv_id TYPE zleave_id,
      lx    TYPE REF TO cx_root.     " exception object reference

DATA: gv_okcode TYPE sy-ucomm.

MODULE status_0100 OUTPUT.
  SET PF-STATUS 'MAIN100'.
  SET TITLEBAR 'LEAVE_MGMT'.
  IF gs_leave_scr-created_by IS INITIAL.
    gs_leave_scr-created_by = sy-uname.
  ENDIF.
ENDMODULE.

MODULE user command 0100 INPUT.
```

output 8



Name	ZELM_LEAVE	Transparent Table
Short Descript.	Employment leave table	
Last Changed	ABAP142	09.11.2025
Status	Actv.	Saved

Logical Storage Parameters

Data Class	APPL0	Master data, transparent tables
Size Category	0	Exected data records 0 to 1.700

Buffering

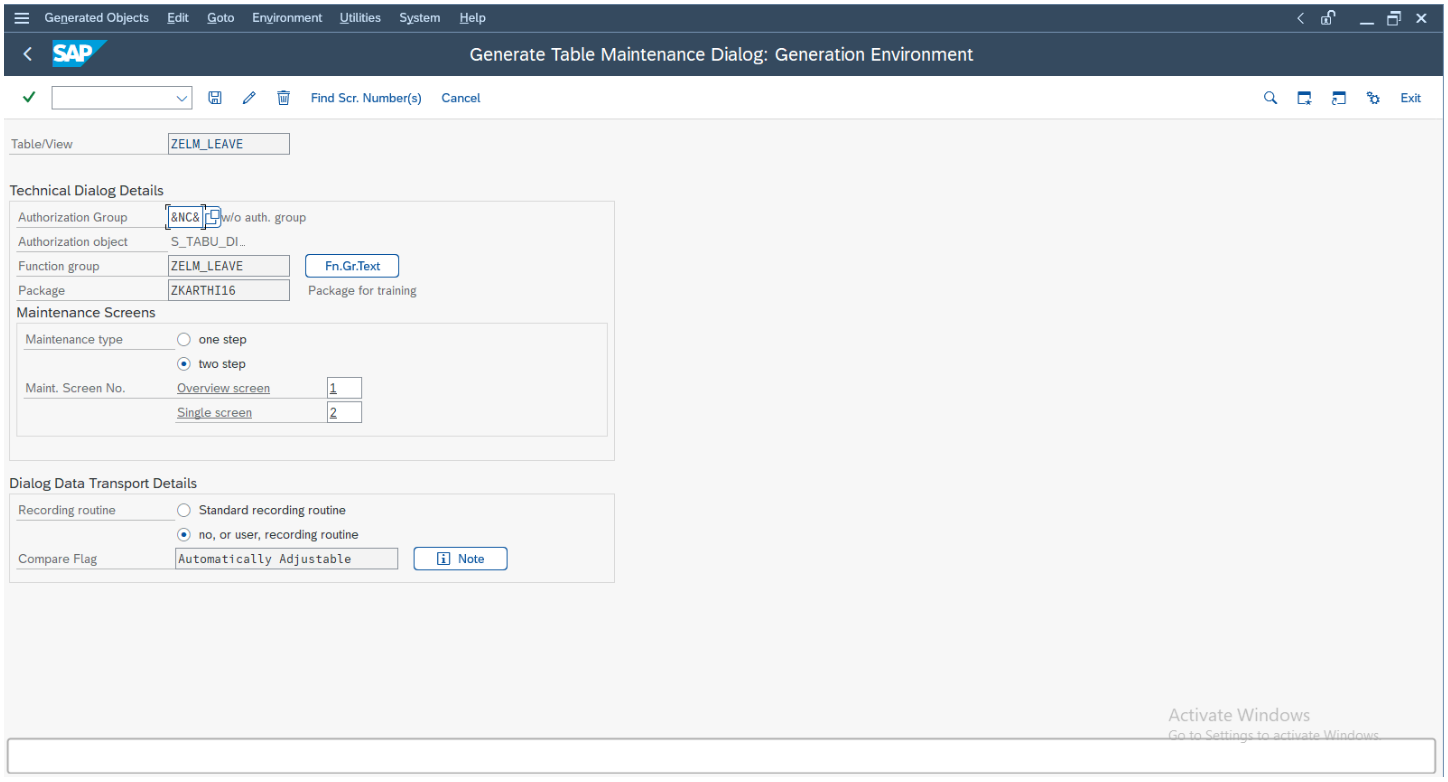
[✓ Buffering Not Allowed]
Buffering allowed but switched off
Buffering Activated

Buffering Type

☐ Single Records Buffered
☐ Generic Area Buffered   Number of Key Fields
☐ Fully Buffered

☐ Log Data Changes
☐ Write access only with JAVA

output 9



Generate Table Maintenance Dialog: Generation Environment

Table/View	ZELM_LEAVE	

Technical Dialog Details

Authorization Group	&NC&	w/o auth. group
Authorization object	S_TABU_DI..	
Function group	ZELM_LEAVE	
Package	ZKARTHI16	Package for training

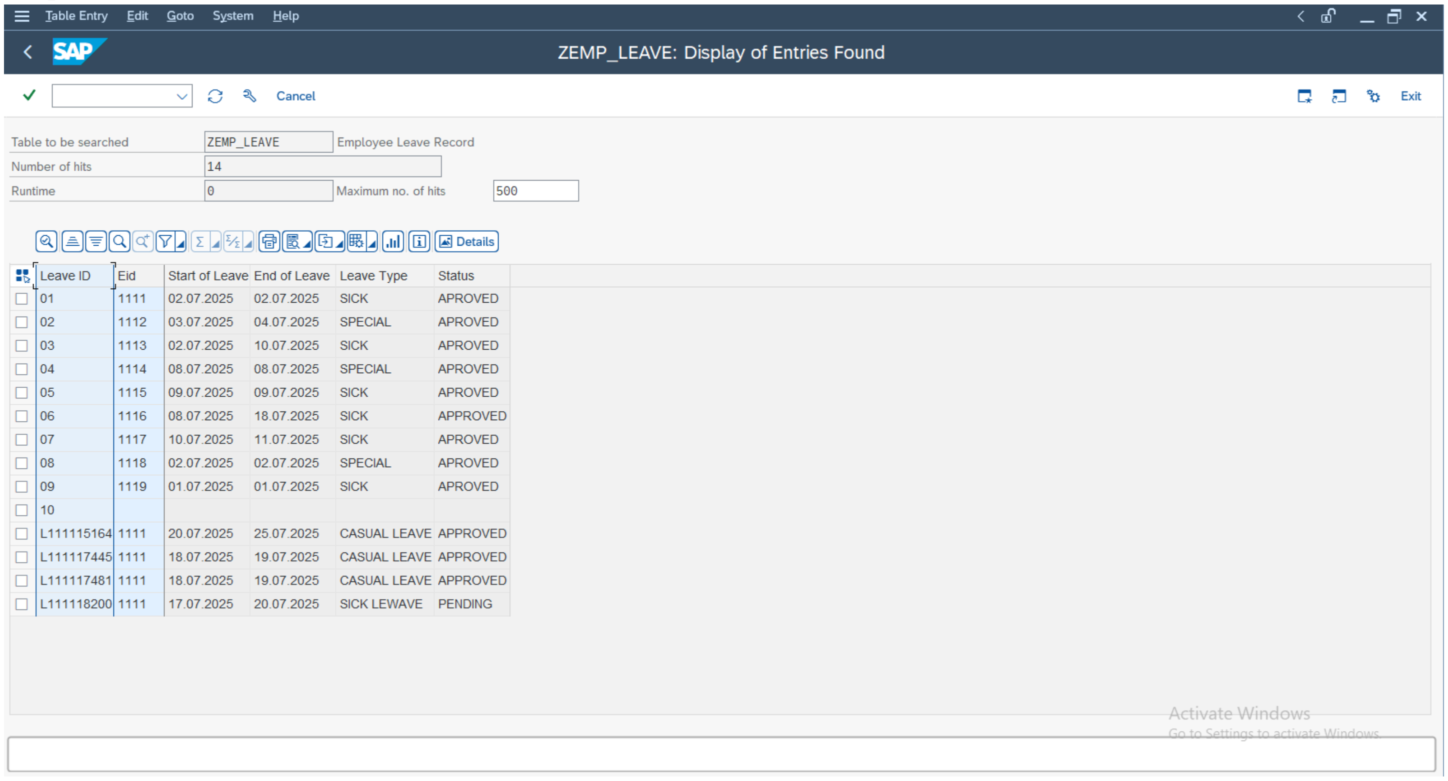
Maintenance Screens

Maintenance type	one step	
	[✓ two step]	
Maint. Screen No.	Overview screen	1
	Single screen	2

Dialog Data Transport Details

Recording routine	Standard recording routine
	[✓ no, or user, recording routine]
Compare Flag	Automatically Adjustable

Output 10.

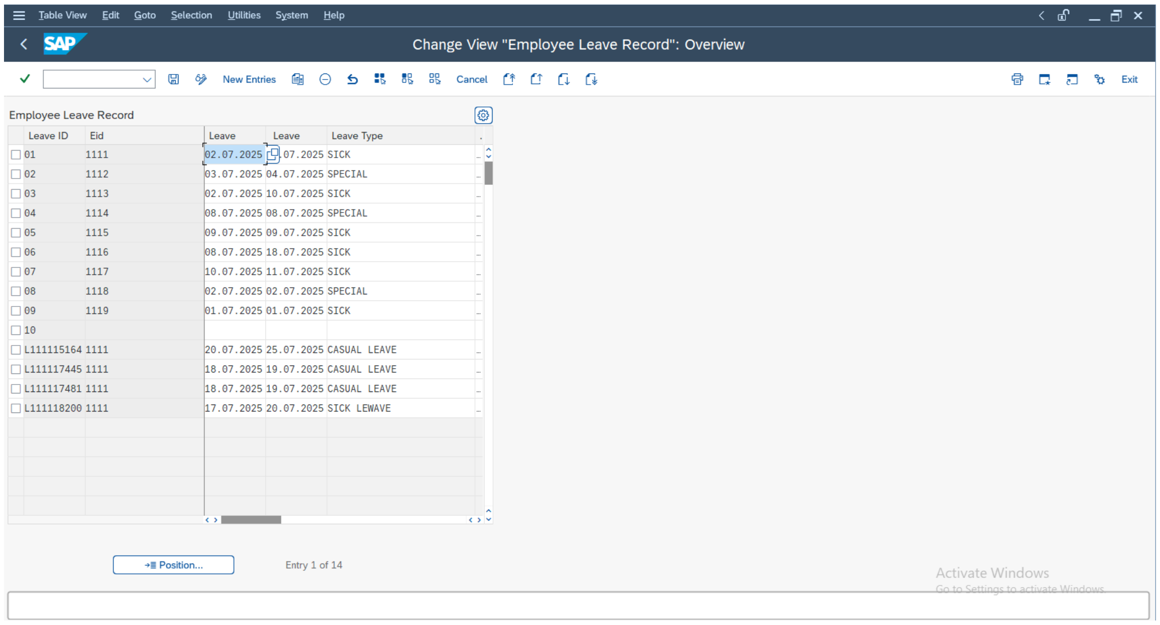


ZEMP_LEAVE: Display of Entries Found

Table to be searched	ZEMP_LEAVE	Employee Leave Record	
Number of hits	14		
Runtime	0	Maximum no. of hits	500

Leave ID	Eid	Start of Leave	End of Leave	Leave Type	Status
01	1111	02.07.2025	02.07.2025	SICK	APROVED
02	1112	03.07.2025	04.07.2025	SPECIAL	APROVED
03	1113	02.07.2025	10.07.2025	SICK	APROVED
04	1114	08.07.2025	08.07.2025	SPECIAL	APROVED
05	1115	09.07.2025	09.07.2025	SICK	APROVED
06	1116	08.07.2025	18.07.2025	SICK	APPROVED
07	1117	10.07.2025	11.07.2025	SICK	APROVED
08	1118	02.07.2025	02.07.2025	SPECIAL	APROVED
09	1119	01.07.2025	01.07.2025	SICK	APROVED
10					
L111115164	1111	20.07.2025	25.07.2025	CASUAL LEAVE	APPROVED
L111117445	1111	18.07.2025	19.07.2025	CASUAL LEAVE	APPROVED
L111117481	1111	18.07.2025	19.07.2025	CASUAL LEAVE	APPROVED
L111118200	1111	17.07.2025	20.07.2025	SICK LEWAVE	PENDING

Output 11



Change View "Employee Leave Record": Overview

Employee Leave Record

Leave ID	Eid	Leave	Leave	Leave Type
01	1111	02.07.2025	.07.2025	SICK
02	1112	03.07.2025	04.07.2025	SPECIAL
03	1113	02.07.2025	10.07.2025	SICK
04	1114	08.07.2025	08.07.2025	SPECIAL
05	1115	09.07.2025	09.07.2025	SICK
06	1116	08.07.2025	18.07.2025	SICK
07	1117	10.07.2025	11.07.2025	SICK
08	1118	02.07.2025	02.07.2025	SPECIAL
09	1119	01.07.2025	01.07.2025	SICK
10				
L111115164	1111	20.07.2025	25.07.2025	CASUAL LEAVE
L111117445	1111	18.07.2025	19.07.2025	CASUAL LEAVE
L111117481	1111	18.07.2025	19.07.2025	CASUAL LEAVE
L111118200	1111	17.07.2025	20.07.2025	SICK LEWAVE

Entry 1 of 14

I) Future Enhancements

- Integration with SAP Smartforms for PDF outputs.
- Workflow-based approval processes.
- Role-based access control for different employee levels.

J) Conclusion

This project successfully demonstrates the implementation of a complete Employee Leave Management System in SAP ABAP.
All essential SAP ABAP components — Data Dictionary, Function Modules, Reports, and Module Pool Programming — have been integrated and tested.
The system provides end-to-end functionality for creating, managing, and reporting employee leave details efficiently.

K) Project Completion Summary

✅ Total Completion: 100%
All modules and functional units have been developed, tested, and validated successfully.
The project fulfills all the academic and practical requirements of a full-cycle ABAP application.