Employee Management System Using SAP-ABAP

Organization: PRIMUS Tech Solutions Pvt Ltd
Project Type: Enterprise Payroll Automation System
Technology: ABAP
Fiscal Year: 2024–25

---

1. Abstract

The Employee Management & Payroll System is designed to automate employee data management and payroll processing for PRIMUS Tech Solutions Pvt Ltd. Organizations with growing workforces face challenges in maintaining accurate payroll calculations, employee records, leave tracking, overtime payments, and statutory deductions.

This project implements a structured payroll system that integrates employee master management, leave accounting, overtime computation, salary processing, and workforce analytics into a single automated framework.

The system calculates salary components including Basic Pay, House Rent Allowance, Dearness Allowance, Conveyance Allowance, and Medical Allowance. It also computes statutory deductions such as Provident Fund, Professional Tax, Employee State Insurance, and Income Tax based on annual tax slabs.

Additional modules generate departmental payroll analysis, annual earnings registers, overtime summaries, leave balance registers, and workforce analytics including gender distribution, salary ranges, and top earners.

The system demonstrates how enterprise payroll logic can be implemented using structured programming concepts such as modular functions, internal tables, and data aggregation techniques.

2. Introduction

Payroll management is a critical function of every organization. Manual payroll processing often results in calculation errors, delayed payments, and compliance risks. Automation improves efficiency, transparency, and accuracy.

This project simulates a corporate payroll management system capable of:

- Maintaining employee master records
- Calculating monthly payroll
- Tracking overtime and leave balances
- Managing promotions and salary increments
- Generating payslips and payroll registers
- Providing workforce analytics and reports

The project demonstrates how structured enterprise logic can be implemented through ABAP programming techniques.

3. Objectives

The main objectives of the project are:

1. Automate payroll calculations.
2. Maintain centralized employee master records.
3. Compute statutory deductions automatically.
4. Track leave balances and overtime payments.
5. Generate detailed payroll registers and payslips.
6. Provide departmental payroll analysis.
7. Generate workforce analytics for management insights.
8. Validate employee data for consistency and accuracy.

4. Scope of the Project

In Scope

- Department master management
- Employee master data management
- Leave tracking
- Overtime management
- Payroll calculation
- Payslip generation
- Annual earnings register
- Workforce analytics

Out of Scope

- Bank integration for salary transfer
- Real-time tax portal integration
- Attendance biometric systems
- Multi-company deployment

5. Methodology

This was developed using a **modular structured programming approach** consisting of:

1. Data Structure Design
2. Business Logic Definition
3. Modular Subroutine Development
4. Sequential Processing Execution
5. Aggregated Reporting

The development process followed three major documentation stages:

- Business Requirements Document (BRD)
- Functional Specification (FS)
- Technical Specification (TS)

6. Business Requirements Document (BRD)

6.1 Business Objective

The organization requires a centralized system to automate employee management and payroll processing while ensuring statutory compliance and generating analytical reports for decision making.

6.2 Stakeholders

Stakeholder	Role
CEO	Strategic oversight
HR Department	Employee management
Finance Department	Payroll and tax compliance
Department Heads	Budget analysis
Employees	Payslip and salary transparency

6.3 Business Requirements

The system must:

1. Maintain department records.
2. Maintain employee master data.
3. Track leave balances.
4. Record overtime hours.
5. Calculate payroll automatically.
6. Generate payslips.
7. Produce annual earnings registers.
8. Perform department payroll analysis.
9. Provide workforce analytics.
10. Validate employee data.

7. Functional Specification (FS)

7.1 System Modules

The system consists of the following functional modules:

1. Department Management
2. Employee Management
3. Leave Management
4. Overtime Management
5. Payroll Processing
6. Bonus Processing
7. Payslip Generation
8. Annual Earnings Register
9. Department Payroll Analysis
10. Workforce Analytics
11. Data Validation
12. System Summary Reporting

7.2 Department Management

The system stores department information including department ID, name, location, head of department, and annual budget. The system calculates employee count and total payroll expenditure for each department.

7.3 Employee Management

The system maintains employee records including employee ID, name, department, designation, salary, grade, gender, and reporting manager. It also handles lifecycle events such as promotions, salary increments, new joiners, and resignations.

7.4 Leave Management

The system tracks the following leave categories:

- Casual Leave
- Medical Leave
- Earned Leave
- Loss of Pay

Leave balances are updated whenever leave is used.

7.5 Overtime Management

The system calculates overtime based on:

- Weekday overtime
- Weekend overtime
- Holiday overtime

Overtime amount depends on employee salary and overtime multiplier.

7.6 Payroll Processing

The system calculates the following salary components:

Earnings

- Basic Salary
- HRA (40%)
- DA (20%)
- Conveyance Allowance
- Medical Allowance

Deductions

- Provident Fund
- Employee State Insurance
- Professional Tax
- Income Tax

Net Pay

Net Pay = Gross Salary – Total Deductions

7.7 Payslip Generation

The system generates structured payslips containing:

- Employee details
- Salary components
- Deduction details
- Net salary

7.8 Workforce Analytics

The system provides analytics including:

- Gender distribution
- Grade distribution
- Average salary
- Highest salary
- Lowest salary
- Top 5 earners

8. Technical Specification (TS)

8.1 System Architecture

The system follows a layered structure:

1. Data Definition Layer
2. Data Storage Layer
3. Processing Layer
4. Aggregation Layer
5. Reporting Layer

8.2 Data Structures

The system defines structured data types for:

- Department records
- Employee records
- Payslip records
- Overtime records
- Leave balances
- Annual earnings register

8.3 Internal Tables

The system uses internal tables to store runtime data:

Table	Purpose
lt_dept	Department master
lt_emp	Employee master
lt_payslip	Payroll records

lt_ot	Overtime records
lt_leave	Leave balances
lt_yreg	Annual earnings register

8.4 Processing Modules

Major system functions include:

- Income tax calculation
- Payslip computation
- Overtime calculation
- Annual earnings aggregation
- Validation engine

8.5 Payroll Calculation Algorithm

1. Calculate HRA and DA percentages.
2. Add fixed allowances.
3. Compute gross salary.
4. Apply statutory deductions.
5. Calculate income tax.
6. Compute net pay.

8.6 Validation Engine

The system validates:

- Salary must be positive
- Grade must be valid
- Salary must match grade band

Errors and warnings are counted and displayed in the validation report.

9. System Outputs

The system generates the following reports:

1. Department Register
2. Employee Directory
3. Payroll Register
4. Payslip Reports
5. Overtime Summary
6. Leave Balance Register
7. Annual Earnings Register
8. Department Payroll Analysis
9. Workforce Analytics
10. Top 5 Earners
11. Validation Report
12. System Execution Summary

10. ABAP Program Implementation

```
from sia.els import els_run
print(els_run("""
*sia
*----------------------------------------------------------------------*
* PROGRAM     : ZEMS_MAIN
* DESCRIPTION : Employee Management & Payroll System
* COMPANY     : Navadurga Tech Solutions Pvt Ltd, Hyderabad
* FISCAL YEAR : 2024-25
*----------------------------------------------------------------------*
*----------------------------------------------------------------------*
* GLOBAL CONSTANTS
*----------------------------------------------------------------------*
CONSTANTS gc_company   TYPE string VALUE 'PRIMUS Tech Solutions Pvt Ltd'.
CONSTANTS gc_gstin     TYPE string VALUE '36AABCN1234F1Z5'.
CONSTANTS gc_city      TYPE string VALUE 'Hyderabad'.
CONSTANTS gc_fy        TYPE string VALUE '2024-25'.
CONSTANTS gc_pf_pct    TYPE i VALUE 12.
CONSTANTS gc_pt        TYPE i VALUE 200.
CONSTANTS gc_ca        TYPE i VALUE 1600.
CONSTANTS gc_ma        TYPE i VALUE 1250.
CONSTANTS gc_cl        TYPE i VALUE 12.
CONSTANTS gc_ml        TYPE i VALUE 15.
CONSTANTS gc_el        TYPE i VALUE 20.
CONSTANTS gc_ot_wd_mul TYPE i VALUE 3.
CONSTANTS gc_ot_we_mul TYPE i VALUE 4.
CONSTANTS gc_ot_hd_mul TYPE i VALUE 6.
*----------------------------------------------------------------------*
* TYPE DEFINITIONS
*----------------------------------------------------------------------*
TYPES BEGIN OF ty_dept.
  deptid   TYPE c LENGTH 6.
  dname    TYPE string.
```

```
  hod      TYPE c LENGTH 5.
  location TYPE string.
  budget   TYPE i.
END OF.
TYPES BEGIN OF ty_emp.
  empno  TYPE c LENGTH 5.
  ename  TYPE string.
  dept   TYPE c LENGTH 6.
  desg   TYPE string.
  basic  TYPE i.
  grade  TYPE c LENGTH 1.
  gender TYPE c LENGTH 1.
  doj    TYPE c LENGTH 8.
  city   TYPE string.
  mgr    TYPE c LENGTH 5.
  status TYPE c LENGTH 1.
END OF.
TYPES BEGIN OF ty_payslip.
  empno    TYPE c LENGTH 5.
  ename    TYPE string.
  dept     TYPE c LENGTH 6.
  desg     TYPE string.
  month    TYPE c LENGTH 6.
  basic    TYPE i.
  hra      TYPE i.
  da       TYPE i.
  ca       TYPE i.
  ma       TYPE i.
  gross    TYPE i.
  pf_emp   TYPE i.
  pf_co    TYPE i.
  esi      TYPE i.
  pt       TYPE i.
  inc_tax  TYPE i.
  tot_ded  TYPE i.
  net_pay  TYPE i.
```

```
END OF.
TYPES BEGIN OF ty_ot.
  empno   TYPE c LENGTH 5.
  month   TYPE c LENGTH 6.
  hrs     TYPE i.
  ot_type TYPE c LENGTH 1.
  amt     TYPE i.
END OF.
TYPES BEGIN OF ty_leave.
  empno    TYPE c LENGTH 5.
  cl_tot   TYPE i.
  cl_used  TYPE i.
  cl_bal   TYPE i.
  ml_tot   TYPE i.
  ml_used  TYPE i.
  ml_bal   TYPE i.
  el_tot   TYPE i.
  el_used  TYPE i.
  el_bal   TYPE i.
  lop      TYPE i.
END OF.
TYPES BEGIN OF ty_yreg.
  empno   TYPE c LENGTH 5.
  ename   TYPE string.
  dept    TYPE c LENGTH 6.
  yr_gross  TYPE i.
  yr_ot     TYPE i.
  yr_bonus  TYPE i.
  yr_pf     TYPE i.
  yr_tax    TYPE i.
  yr_net    TYPE i.
END OF.
*---------------------------------------------------------------------*
* INTERNAL TABLES
*---------------------------------------------------------------------*
DATA lt_dept    TYPE TABLE OF ty_dept.
```

```
DATA lt_emp     TYPE TABLE OF ty_emp.
DATA lt_payslip TYPE TABLE OF ty_payslip.
DATA lt_ot      TYPE TABLE OF ty_ot.
DATA lt_leave   TYPE TABLE OF ty_leave.
DATA lt_yreg    TYPE TABLE OF ty_yreg.
*----------------------------------------------------------------------*
* WORK AREAS
*----------------------------------------------------------------------*
DATA wa_dept    TYPE ty_dept.
DATA wa_emp     TYPE ty_emp.
DATA wa_payslip TYPE ty_payslip.
DATA wa_ot      TYPE ty_ot.
DATA wa_leave   TYPE ty_leave.
DATA wa_yreg    TYPE ty_yreg.
*----------------------------------------------------------------------*
* WORKING VARIABLES
*----------------------------------------------------------------------*
DATA lv_i       TYPE i.
DATA lv_month   TYPE c LENGTH 6.
DATA lv_line    TYPE string.
DATA lv_count   TYPE i.
DATA lv_total   TYPE i.
DATA lv_avg     TYPE i.
DATA lv_temp    TYPE i.
DATA lv_bonus   TYPE i.
DATA lv_ot_amt  TYPE i.
DATA lv_found   TYPE i.
DATA lv_ot_W    TYPE i.
DATA lv_ot_E    TYPE i.
DATA lv_ot_H    TYPE i.
*----------------------------------------------------------------------*
* FORM: F_INCOME_TAX
* Calculate monthly income tax from annual gross
*----------------------------------------------------------------------*
FORM f_income_tax USING p_gross CHANGING p_tax.
  DATA lv_ann TYPE i.
```

```
  lv_ann = p_gross * 12.
  IF lv_ann > 1000000.
    p_tax = ( ( lv_ann - 1000000 ) * 20 / 100 ) + 37500.
  ELSEIF lv_ann > 750000.
    p_tax = ( ( lv_ann - 750000 ) * 15 / 100 ) + 25000.
  ELSEIF lv_ann > 500000.
    p_tax = ( ( lv_ann - 500000 ) * 10 / 100 ) + 12500.
  ELSEIF lv_ann > 250000.
    p_tax = ( lv_ann - 250000 ) * 5 / 100.
  ELSE.
    p_tax = 0.
  ENDIF.
  p_tax = p_tax / 12.
ENDFORM.
*----------------------------------------------------------------------*
* FORM: F_CALC_PAYSLIP
* Populate all payslip fields from basic salary
*----------------------------------------------------------------------*
FORM f_calc_payslip USING p_month CHANGING p_wa TYPE ty_payslip.
  DATA lv_tax TYPE i.
  p_wa-month   = p_month.
  p_wa-hra     = p_wa-basic * 40 / 100.
  p_wa-da      = p_wa-basic * 20 / 100.
  p_wa-ca      = gc_ca.
  p_wa-ma      = gc_ma.
  p_wa-gross   = p_wa-basic + p_wa-hra + p_wa-da + p_wa-ca + p_wa-ma.
  p_wa-pf_emp  = p_wa-basic * gc_pf_pct / 100.
  p_wa-pf_co   = p_wa-pf_emp.
  IF p_wa-gross <= 21000.
    p_wa-esi = p_wa-gross * 75 / 10000.
  ELSE.
    p_wa-esi = 0.
  ENDIF.
  p_wa-pt      = gc_pt.
  PERFORM f_income_tax USING p_wa-gross CHANGING lv_tax.
  p_wa-inc_tax = lv_tax.
```

```
  p_wa-tot_ded = p_wa-pf_emp + p_wa-esi + p_wa-pt + p_wa-inc_tax.
  p_wa-net_pay = p_wa-gross - p_wa-tot_ded.
ENDFORM.
*----------------------------------------------------------------------*
* FORM: F_CALC_OT
* OT amount based on type: W=weekday E=weekend H=holiday
*----------------------------------------------------------------------*
FORM f_calc_ot USING p_basic p_hrs p_type CHANGING p_amt.
  DATA lv_hr TYPE i.
  lv_hr = p_basic / 26 / 8.
  CASE p_type.
    WHEN 'W'. p_amt = p_hrs * lv_hr * gc_ot_wd_mul / 2.
    WHEN 'E'. p_amt = p_hrs * lv_hr * gc_ot_we_mul / 2.
    WHEN 'H'. p_amt = p_hrs * lv_hr * gc_ot_hd_mul / 2.
    WHEN OTHERS. p_amt = 0.
  ENDCASE.
ENDFORM.
*----------------------------------------------------------------------*
* FORM: F_PRINT_PAYSLIP
* Print formatted payslip for one employee for a given month
*----------------------------------------------------------------------*
FORM f_print_payslip USING p_empno p_month.
  DATA wa_e  TYPE ty_emp.
  DATA wa_ps TYPE ty_payslip.
  DATA lv_l  TYPE string.
  READ TABLE lt_emp INTO wa_e WITH KEY empno = p_empno.
  IF sy-subrc <> 0.
    WRITE / '  [ERROR] Employee not found.'.
  ELSE.
    wa_ps-empno = wa_e-empno.
    wa_ps-ename = wa_e-ename.
    wa_ps-dept  = wa_e-dept.
    wa_ps-desg  = wa_e-desg.
    wa_ps-basic = wa_e-basic.
    PERFORM f_calc_payslip USING p_month CHANGING wa_ps.
    WRITE / '  +-----------------------------------------------------------
```

```
+'.
    CONCATENATE '  | ' gc_company INTO lv_l.
    WRITE / lv_l.
    WRITE / '  | PAYSLIP'.
    WRITE / '  +------------------------------------------------------------
+'.
    CONCATENATE '  | Emp No      : ' wa_ps-empno INTO lv_l.
    WRITE / lv_l.
    CONCATENATE '  | Name        : ' wa_ps-ename INTO lv_l.
    WRITE / lv_l.
    CONCATENATE '  | Department : ' wa_ps-dept INTO lv_l.
    WRITE / lv_l.
    CONCATENATE '  | Designation: ' wa_ps-desg INTO lv_l.
    WRITE / lv_l.
    CONCATENATE '  | Pay Period : ' p_month INTO lv_l.
    WRITE / lv_l.
    WRITE / '  +-----------------------------+------------------------------
+'.
    WRITE / '  |         EARNINGS            |         DEDUCTIONS
|'.
    WRITE / '  +-----------------------------+------------------------------
+'.
    CONCATENATE '  | Basic Salary    : ' wa_ps-basic   INTO lv_l. WRITE /
lv_l.
    CONCATENATE '  | HRA        (40%): ' wa_ps-hra     INTO lv_l. WRITE /
lv_l.
    CONCATENATE '  | Dearness   (20%): ' wa_ps-da      INTO lv_l. WRITE /
lv_l.
    CONCATENATE '  | Conveyance      : ' wa_ps-ca      INTO lv_l. WRITE /
lv_l.
    CONCATENATE '  | Medical Allow   : ' wa_ps-ma      INTO lv_l. WRITE /
lv_l.
    CONCATENATE '  |                     | PF (Employee): ' wa_ps-pf_emp
INTO lv_l. WRITE / lv_l.
    CONCATENATE '  |                     | PF (Employer): ' wa_ps-pf_co
INTO lv_l. WRITE / lv_l.
```

```
    CONCATENATE '  |                         | Prof Tax      : ' wa_ps-pt
INTO lv_l. WRITE / lv_l.
    CONCATENATE '  |                         | Income Tax    : ' wa_ps-inc_tax
INTO lv_l. WRITE / lv_l.
    CONCATENATE '  |                         | ESI           : ' wa_ps-esi
INTO lv_l. WRITE / lv_l.
    WRITE / '  +----------------------------+----------------------------
+'.
    CONCATENATE '  | Gross Salary    : ' wa_ps-gross   INTO lv_l. WRITE /
lv_l.
    CONCATENATE '  | Total Deductions: ' wa_ps-tot_ded INTO lv_l. WRITE /
lv_l.
    WRITE / '  +---------------------------------------------------------
+'.
    CONCATENATE '  |  NET PAY  :  INR ' wa_ps-net_pay INTO lv_l.
    WRITE / lv_l.
    WRITE / '  +---------------------------------------------------------
+'.
    WRITE / ' '.
  ENDIF.
ENDFORM.
*-----------------------------------------------------------------------*
* MASTER DATA LOAD - DEPARTMENTS
*-----------------------------------------------------------------------*
APPEND VALUE #( deptid='IT'    dname='Information Technology'   hod='E003'
location='Block A, Floor 3'  budget=8000000 ) TO lt_dept.
APPEND VALUE #( deptid='HR'    dname='Human Resources'          hod='E002'
location='Block B, Floor 1'  budget=3000000 ) TO lt_dept.
APPEND VALUE #( deptid='FIN'   dname='Finance & Accounts'       hod='E004'
location='Block B, Floor 2'  budget=4500000 ) TO lt_dept.
APPEND VALUE #( deptid='OPS'   dname='Operations'               hod='E007'
location='Block C, Floor 1'  budget=5000000 ) TO lt_dept.
APPEND VALUE #( deptid='SALES' dname='Sales & Marketing'        hod='E009'
location='Block C, Floor 2'  budget=7000000 ) TO lt_dept.
APPEND VALUE #( deptid='ADMIN' dname='Administration'           hod='E012'
location='Block A, Floor 1'  budget=32000000 ) TO lt_dept.
```

```
*----------------------------------------------------------------------*
* MASTER DATA LOAD - EMPLOYEES
*----------------------------------------------------------------------*
APPEND VALUE #( empno='E001' ename='Ravi Kumar'       dept='IT'
desg='Software Engineer'         basic=45000  grade='B' gender='M'
doj='20200115' city='Hyderabad'    mgr='E003' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E002' ename='Priya Sharma'     dept='HR'
desg='HR Manager'                basic=58000  grade='A' gender='F'
doj='20190601' city='Hyderabad'    mgr='E015' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E003' ename='Suresh Reddy'     dept='IT'
desg='Tech Lead'                 basic=78000  grade='A' gender='M'
doj='20180301' city='Hyderabad'    mgr='E015' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E004' ename='Deepa Nair'       dept='FIN'
desg='Finance Manager'           basic=68000  grade='A' gender='F'
doj='20210901' city='Bangalore'    mgr='E015' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E005' ename='Kiran Rao'        dept='IT'
desg='Senior Developer'          basic=65000  grade='A' gender='M'
doj='20170601' city='Secunderabad' mgr='E003' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E006' ename='Meena Patel'      dept='HR'
desg='HR Executive'              basic=36000  grade='C' gender='F'
doj='20220101' city='Hyderabad'    mgr='E002' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E007' ename='Srinivas Kumar'   dept='OPS'
desg='Operations Manager'        basic=60000  grade='A' gender='M'
doj='20160401' city='Hyderabad'    mgr='E015' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E008' ename='Anitha Reddy'     dept='IT'
desg='Software Engineer'         basic=44000  grade='B' gender='F'
doj='20210301' city='Hyderabad'    mgr='E003' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E009' ename='Arjun Mehta'      dept='SALES'
desg='Sales Manager'             basic=62000  grade='A' gender='M'
doj='20150901' city='Mumbai'       mgr='E015' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E010' ename='Lakshmi Devi'     dept='OPS'
desg='Operations Executive'      basic=34000  grade='C' gender='F'
doj='20220801' city='Hyderabad'    mgr='E007' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E011' ename='Rohit Sharma'     dept='IT'
desg='Junior Developer'          basic=32000  grade='C' gender='M'
doj='20230101' city='Hyderabad'    mgr='E003' status='A' ) TO lt_emp.
```

```
APPEND VALUE #( empno='E012' ename='Rajesh Babu'      dept='ADMIN'
desg='Admin Manager'              basic=44000  grade='B' gender='M'
doj='20181201' city='Hyderabad'    mgr='E015' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E013' ename='Venkat Rao'       dept='FIN'
desg='Senior Accountant'          basic=40000  grade='B' gender='M'
doj='20200601' city='Hyderabad'    mgr='E004' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E014' ename='Pooja Singh'      dept='SALES'
desg='Sales Executive'            basic=30000  grade='C' gender='F'
doj='20230601' city='Hyderabad'    mgr='E009' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E015' ename='Ram S'            dept='ADMIN'
desg='Chief Executive Officer'   basic=220000 grade='S' gender='M'
doj='20100601' city='Hyderabad'    mgr=''     status='A' ) TO lt_emp.
APPEND VALUE #( empno='E016' ename='Divya Krishnan'   dept='IT'
desg='Software Engineer'          basic=38000  grade='C' gender='F'
doj='20240301' city='Hyderabad'    mgr='E003' status='A' ) TO lt_emp.
APPEND VALUE #( empno='E017' ename='Manoj Tiwari'     dept='SALES'
desg='Business Dev Executive'    basic=29000  grade='C' gender='M'
doj='20240401' city='Delhi'        mgr='E009' status='A' ) TO lt_emp.
*---------------------------------------------------------------------*
* INITIALISE LEAVE BALANCES
*---------------------------------------------------------------------*
LOOP AT lt_emp INTO wa_emp.
  wa_leave-empno   = wa_emp-empno.
  wa_leave-cl_tot  = gc_cl.
  wa_leave-cl_used = 0.
  wa_leave-cl_bal  = gc_cl.
  wa_leave-ml_tot  = gc_ml.
  wa_leave-ml_used = 0.
  wa_leave-ml_bal  = gc_ml.
  wa_leave-el_tot  = gc_el.
  wa_leave-el_used = 0.
  wa_leave-el_bal  = gc_el.
  wa_leave-lop     = 0.
  APPEND wa_leave TO lt_leave.
ENDLOOP.
*---------------------------------------------------------------------*
```

```
* SIMULATE LEAVE USAGE - JAN to AUG 2024
*----------------------------------------------------------------------*
READ TABLE lt_leave INTO wa_leave WITH KEY empno = 'E001'.
wa_leave-cl_used = 3.  wa_leave-cl_bal = gc_cl - 3.
wa_leave-ml_used = 2.  wa_leave-ml_bal = gc_ml - 2.
MODIFY TABLE lt_leave FROM wa_leave.
READ TABLE lt_leave INTO wa_leave WITH KEY empno = 'E005'.
wa_leave-cl_used = 5.  wa_leave-cl_bal = gc_cl - 5.
wa_leave-el_used = 4.  wa_leave-el_bal = gc_el - 4.
MODIFY TABLE lt_leave FROM wa_leave.
READ TABLE lt_leave INTO wa_leave WITH KEY empno = 'E008'.
wa_leave-ml_used = 7.  wa_leave-ml_bal = gc_ml - 7.
wa_leave-lop     = 1.
MODIFY TABLE lt_leave FROM wa_leave.
READ TABLE lt_leave INTO wa_leave WITH KEY empno = 'E011'.
wa_leave-cl_used = 4.  wa_leave-cl_bal = gc_cl - 4.
wa_leave-ml_used = 3.  wa_leave-ml_bal = gc_ml - 3.
MODIFY TABLE lt_leave FROM wa_leave.
READ TABLE lt_leave INTO wa_leave WITH KEY empno = 'E014'.
wa_leave-cl_used = 2.  wa_leave-cl_bal = gc_cl - 2.
wa_leave-lop     = 2.
MODIFY TABLE lt_leave FROM wa_leave.
*----------------------------------------------------------------------*
* GENERATE OT RECORDS - GRADE A AND B, ALL 12 MONTHS
*----------------------------------------------------------------------*
DO 12 TIMES.
  lv_i = sy-index.
  IF lv_i < 10.
    CONCATENATE '20240' lv_i INTO lv_month.
  ELSE.
    CONCATENATE '2024' lv_i INTO lv_month.
  ENDIF.
  LOOP AT lt_emp INTO wa_emp.
    CHECK wa_emp-grade = 'A' OR wa_emp-grade = 'B'.
    IF lv_i = 3 OR lv_i = 9.
      wa_ot-hrs     = 20.
```

```
      wa_ot-ot_type = 'W'.
    ELSEIF lv_i = 6 OR lv_i = 12.
      wa_ot-hrs     = 25.
      wa_ot-ot_type = 'E'.
    ELSEIF lv_i = 10 OR lv_i = 11.
      wa_ot-hrs     = 12.
      wa_ot-ot_type = 'H'.
    ELSE.
      wa_ot-hrs     = 8.
      wa_ot-ot_type = 'W'.
    ENDIF.
    wa_ot-empno = wa_emp-empno.
    wa_ot-month = lv_month.
    wa_ot-amt   = 0.
    PERFORM f_calc_ot USING wa_emp-basic wa_ot-hrs wa_ot-ot_type CHANGING
wa_ot-amt.
    APPEND wa_ot TO lt_ot.
  ENDLOOP.
ENDDO.
*----------------------------------------------------------------------*
* MID-YEAR HR EVENTS
* New joiner, promotions, annual increment, resignation
*----------------------------------------------------------------------*
* New joiner Sep 2024
APPEND VALUE #( empno='E018' ename='Shalini Verma'  dept='IT'
desg='Software Engineer'   basic=40000  grade='C' gender='F'
doj='20240901' city='Hyderabad' mgr='E003' status='A' ) TO lt_emp.
READ TABLE lt_leave INTO wa_leave WITH KEY empno = 'E018'.
IF sy-subrc <> 0.
  wa_leave-empno   = 'E018'.
  wa_leave-cl_tot  = gc_cl.  wa_leave-cl_used = 0.  wa_leave-cl_bal =
gc_cl.
  wa_leave-ml_tot  = gc_ml.  wa_leave-ml_used = 0.  wa_leave-ml_bal =
gc_ml.
  wa_leave-el_tot  = gc_el.  wa_leave-el_used = 0.  wa_leave-el_bal =
gc_el.
```

```
  wa_leave-lop     = 0.
  APPEND wa_leave TO lt_leave.
ENDIF.
* Annual increment - all Grade C get 8%
LOOP AT lt_emp INTO wa_emp WHERE grade = 'C'.
  lv_temp = wa_emp-basic * 8 / 100.
  ADD lv_temp TO wa_emp-basic.
  MODIFY TABLE lt_emp FROM wa_emp.
ENDLOOP.
* Promotion: E001 Ravi Kumar  B -> A, 20% hike
READ TABLE lt_emp INTO wa_emp WITH KEY empno = 'E001'.
IF sy-subrc = 0.
  lv_temp = wa_emp-basic * 20 / 100.
  ADD lv_temp TO wa_emp-basic.
  wa_emp-grade = 'A'.
  wa_emp-desg  = 'Senior Software Engineer'.
  MODIFY TABLE lt_emp FROM wa_emp.
ENDIF.
* Promotion: E008 Anitha Reddy  B -> A, 18% hike
READ TABLE lt_emp INTO wa_emp WITH KEY empno = 'E008'.
IF sy-subrc = 0.
  lv_temp = wa_emp-basic * 18 / 100.
  ADD lv_temp TO wa_emp-basic.
  wa_emp-grade = 'A'.
  wa_emp-desg  = 'Lead Engineer'.
  MODIFY TABLE lt_emp FROM wa_emp.
ENDIF.
* Resignation: E006 Meena Patel - Sep 2024
DELETE TABLE lt_emp   WITH KEY empno = 'E006'.
DELETE TABLE lt_leave WITH KEY empno = 'E006'.
*----------------------------------------------------------------------*
* PAYROLL RUN - JANUARY 2024 (BASE MONTH DETAILED REGISTER)
*----------------------------------------------------------------------*
LOOP AT lt_emp INTO wa_emp.
  wa_payslip-empno = wa_emp-empno.
  wa_payslip-ename = wa_emp-ename.
```

```
  wa_payslip-dept  = wa_emp-dept.
  wa_payslip-desg  = wa_emp-desg.
  wa_payslip-basic = wa_emp-basic.
  PERFORM f_calc_payslip USING '202401' CHANGING wa_payslip.
  APPEND wa_payslip TO lt_payslip.
ENDLOOP.
*----------------------------------------------------------------------*
* YEARLY REGISTER BUILD - ALL 12 MONTHS x ALL EMPLOYEES
*----------------------------------------------------------------------*
DO 12 TIMES.
  lv_i = sy-index.
  IF lv_i < 10.
    CONCATENATE '20240' lv_i INTO lv_month.
  ELSE.
    CONCATENATE '2024' lv_i INTO lv_month.
  ENDIF.
  LOOP AT lt_emp INTO wa_emp.
    wa_payslip-empno = wa_emp-empno.
    wa_payslip-ename = wa_emp-ename.
    wa_payslip-dept  = wa_emp-dept.
    wa_payslip-desg  = wa_emp-desg.
    wa_payslip-basic = wa_emp-basic.
    PERFORM f_calc_payslip USING lv_month CHANGING wa_payslip.
    * Festival bonus: Oct = 10%, Nov = 15% of basic
    lv_bonus = 0.
    IF lv_i = 10.
      lv_bonus = wa_emp-basic * 10 / 100.
    ELSEIF lv_i = 11.
      lv_bonus = wa_emp-basic * 15 / 100.
    ELSE.
      lv_bonus = 0.
    ENDIF.
    * OT for this employee this month
    lv_ot_amt = 0.
    LOOP AT lt_ot INTO wa_ot.
      CHECK wa_ot-empno = wa_emp-empno.
```

```
      CHECK wa_ot-month = lv_month.
      lv_ot_amt = wa_ot-amt.
    ENDLOOP.
    * Accumulate into yearly register
    lv_found = 0.
    LOOP AT lt_yreg INTO wa_yreg.
      IF wa_yreg-empno = wa_emp-empno.
        lv_found = 1.
        ADD wa_payslip-gross  TO wa_yreg-yr_gross.
        ADD lv_ot_amt         TO wa_yreg-yr_ot.
        ADD lv_bonus          TO wa_yreg-yr_bonus.
        ADD wa_payslip-pf_emp TO wa_yreg-yr_pf.
        ADD wa_payslip-inc_tax TO wa_yreg-yr_tax.
        lv_temp = wa_payslip-net_pay + lv_ot_amt + lv_bonus.
        ADD lv_temp TO wa_yreg-yr_net.
        MODIFY TABLE lt_yreg FROM wa_yreg.
      ENDIF.
    ENDLOOP.
    IF lv_found = 0.
      wa_yreg-empno    = wa_emp-empno.
      wa_yreg-ename    = wa_emp-ename.
      wa_yreg-dept     = wa_emp-dept.
      wa_yreg-yr_gross = wa_payslip-gross.
      wa_yreg-yr_ot    = lv_ot_amt.
      wa_yreg-yr_bonus = lv_bonus.
      wa_yreg-yr_pf    = wa_payslip-pf_emp.
      wa_yreg-yr_tax   = wa_payslip-inc_tax.
      wa_yreg-yr_net   = wa_payslip-net_pay + lv_ot_amt + lv_bonus.
      APPEND wa_yreg TO lt_yreg.
    ENDIF.
  ENDLOOP.
ENDDO.
*----------------------------------------------------------------------*
* OUTPUT: COMPANY HEADER
*----------------------------------------------------------------------*
WRITE / ' '.
```

```
WRITE /
'================================================================'.
WRITE / gc_company.
WRITE / gc_city.
WRITE / 'EMPLOYEE MANAGEMENT & PAYROLL SYSTEM'.
WRITE / 'Fiscal Year:'.
WRITE / gc_fy.
WRITE /
'================================================================'.
*----------------------------------------------------------------------*
* OUTPUT: DEPARTMENT REGISTER
*----------------------------------------------------------------------*
WRITE / ' '.
WRITE / '--- DEPARTMENT REGISTER ---'.
LOOP AT lt_dept INTO wa_dept.
  lv_count = 0.
  lv_total = 0.
  LOOP AT lt_emp INTO wa_emp WHERE dept = wa_dept-deptid.
    ADD 1 TO lv_count.
    ADD wa_emp-basic TO lv_total.
  ENDLOOP.
  WRITE / wa_dept-deptid.
  WRITE / wa_dept-dname.
  WRITE / wa_dept-location.
  WRITE / lv_count.
  WRITE / lv_total.
ENDLOOP.
*----------------------------------------------------------------------*
* OUTPUT: EMPLOYEE DIRECTORY
*----------------------------------------------------------------------*
WRITE / ' '.
WRITE / '--- EMPLOYEE DIRECTORY ---'.
LOOP AT lt_emp INTO wa_emp.
  WRITE / wa_emp-empno.
  WRITE / wa_emp-ename.
  WRITE / wa_emp-dept.
```

```
  WRITE / wa_emp-desg.
  WRITE / wa_emp-grade.
  WRITE / wa_emp-basic.
  WRITE / wa_emp-doj.
ENDLOOP.
*----------------------------------------------------------------------*
* OUTPUT: JANUARY 2024 PAYROLL REGISTER
*----------------------------------------------------------------------*
DATA lv_jan_gross TYPE i VALUE 0.
DATA lv_jan_ded   TYPE i VALUE 0.
DATA lv_jan_net   TYPE i VALUE 0.
DATA lv_jan_pf    TYPE i VALUE 0.
DATA lv_jan_tax   TYPE i VALUE 0.
LOOP AT lt_payslip INTO wa_payslip.
  ADD wa_payslip-gross   TO lv_jan_gross.
  ADD wa_payslip-tot_ded TO lv_jan_ded.
  ADD wa_payslip-net_pay TO lv_jan_net.
  ADD wa_payslip-pf_emp  TO lv_jan_pf.
  ADD wa_payslip-inc_tax TO lv_jan_tax.
ENDLOOP.
WRITE / ' '.
WRITE / '--- PAYROLL REGISTER: JANUARY 2024 ---'.
LOOP AT lt_payslip INTO wa_payslip.
  WRITE / wa_payslip-empno.
  WRITE / wa_payslip-ename.
  WRITE / wa_payslip-basic.
  WRITE / wa_payslip-gross.
  WRITE / wa_payslip-tot_ded.
  WRITE / wa_payslip-net_pay.
ENDLOOP.
WRITE / 'Total Gross :'.
WRITE / lv_jan_gross.
WRITE / 'Total Deductions :'.
WRITE / lv_jan_ded.
WRITE / 'Total Net Pay :'.
WRITE / lv_jan_net.
```

```
WRITE / 'Total PF Collected :'.
WRITE / lv_jan_pf.
WRITE / 'Total Income Tax :'.
WRITE / lv_jan_tax.
*----------------------------------------------------------------------*
* OUTPUT: INDIVIDUAL PAYSLIPS (KEY EMPLOYEES)
*----------------------------------------------------------------------*
WRITE / ' '.
WRITE / '--- PAYSLIPS: MARCH 2025 ---'.
PERFORM f_print_payslip USING 'E015' '202503'.
PERFORM f_print_payslip USING 'E003' '202503'.
PERFORM f_print_payslip USING 'E001' '202503'.
PERFORM f_print_payslip USING 'E018' '202503'.
*----------------------------------------------------------------------*
* OUTPUT: OT SUMMARY
*----------------------------------------------------------------------*
DATA lv_ot_tot  TYPE i VALUE 0.
DATA lv_ot_recs TYPE i VALUE 0.
lv_ot_W = 0.
lv_ot_E = 0.
lv_ot_H = 0.
LOOP AT lt_ot INTO wa_ot.
  ADD 1 TO lv_ot_recs.
  ADD wa_ot-amt TO lv_ot_tot.
  CASE wa_ot-ot_type.
    WHEN 'W'. ADD wa_ot-amt TO lv_ot_W.
    WHEN 'E'. ADD wa_ot-amt TO lv_ot_E.
    WHEN 'H'. ADD wa_ot-amt TO lv_ot_H.
  ENDCASE.
ENDLOOP.
WRITE / ' '.
WRITE / '--- OVERTIME SUMMARY: FY 2024-25 ---'.
WRITE / 'Total OT Records :'.
WRITE / lv_ot_recs.
WRITE / 'Weekday OT Amount :'.
WRITE / lv_ot_W.
```

```
WRITE / 'Weekend OT Amount :'.
WRITE / lv_ot_E.
WRITE / 'Holiday OT Amount :'.
WRITE / lv_ot_H.
WRITE / 'Total OT Disbursed :'.
WRITE / lv_ot_tot.
*---------------------------------------------------------------------*
* OUTPUT: LEAVE BALANCE REGISTER
*---------------------------------------------------------------------*
DATA lv_lop_total TYPE i VALUE 0.
WRITE / ' '.
WRITE / '--- LEAVE BALANCE REGISTER: FY 2024 ---'.
LOOP AT lt_leave INTO wa_leave.
  ADD wa_leave-lop TO lv_lop_total.
  WRITE / wa_leave-empno.
  WRITE / wa_leave-cl_used.
  WRITE / wa_leave-cl_bal.
  WRITE / wa_leave-ml_used.
  WRITE / wa_leave-ml_bal.
  WRITE / wa_leave-el_bal.
  WRITE / wa_leave-lop.
ENDLOOP.
WRITE / 'Total LOP Days Across Company :'.
WRITE / lv_lop_total.
*---------------------------------------------------------------------*
* OUTPUT: ANNUAL EARNINGS REGISTER
*---------------------------------------------------------------------*
DATA lv_yr_gross TYPE i VALUE 0.
DATA lv_yr_ot    TYPE i VALUE 0.
DATA lv_yr_bon   TYPE i VALUE 0.
DATA lv_yr_pf    TYPE i VALUE 0.
DATA lv_yr_tax   TYPE i VALUE 0.
DATA lv_yr_net   TYPE i VALUE 0.
LOOP AT lt_yreg INTO wa_yreg.
  ADD wa_yreg-yr_gross  TO lv_yr_gross.
  ADD wa_yreg-yr_ot     TO lv_yr_ot.
```

```
  ADD wa_yreg-yr_bonus  TO lv_yr_bon.
  ADD wa_yreg-yr_pf     TO lv_yr_pf.
  ADD wa_yreg-yr_tax    TO lv_yr_tax.
  ADD wa_yreg-yr_net    TO lv_yr_net.
ENDLOOP.
WRITE / ' '.
WRITE / '--- ANNUAL EARNINGS REGISTER: FY 2024-25 ---'.
LOOP AT lt_yreg INTO wa_yreg.
  WRITE / wa_yreg-empno.
  WRITE / wa_yreg-ename.
  WRITE / wa_yreg-dept.
  WRITE / wa_yreg-yr_gross.
  WRITE / wa_yreg-yr_ot.
  WRITE / wa_yreg-yr_bonus.
  WRITE / wa_yreg-yr_pf.
  WRITE / wa_yreg-yr_tax.
  WRITE / wa_yreg-yr_net.
ENDLOOP.
WRITE / 'Annual Gross Payroll :'.
WRITE / lv_yr_gross.
WRITE / 'Annual OT Disbursed :'.
WRITE / lv_yr_ot.
WRITE / 'Festival Bonus :'.
WRITE / lv_yr_bon.
WRITE / 'PF Collected :'.
WRITE / lv_yr_pf.
WRITE / 'Tax Deducted :'.
WRITE / lv_yr_tax.
WRITE / 'Net Annual Disbursement :'.
WRITE / lv_yr_net.
*----------------------------------------------------------------------*
* OUTPUT: DEPARTMENT-WISE PAYROLL ANALYSIS
*----------------------------------------------------------------------*
DATA lv_d_cnt  TYPE i.
DATA lv_d_pay  TYPE i.
DATA lv_d_net  TYPE i.
```

```
DATA lv_d_budget TYPE i.
DATA lv_d_var  TYPE i.
WRITE / ' '.
WRITE / '--- DEPARTMENT PAYROLL ANALYSIS ---'.
LOOP AT lt_dept INTO wa_dept.
  lv_d_cnt = 0.
  lv_d_pay = 0.
  lv_d_net = 0.
  LOOP AT lt_yreg INTO wa_yreg WHERE dept = wa_dept-deptid.
    ADD 1 TO lv_d_cnt.
    ADD wa_yreg-yr_gross TO lv_d_pay.
    ADD wa_yreg-yr_net   TO lv_d_net.
  ENDLOOP.
  lv_d_budget = wa_dept-budget.
  lv_d_var    = lv_d_budget - lv_d_pay.
  WRITE / wa_dept-deptid.
  WRITE / wa_dept-dname.
  WRITE / lv_d_cnt.
  WRITE / lv_d_pay.
  WRITE / lv_d_net.
  WRITE / lv_d_budget.
  WRITE / lv_d_var.
ENDLOOP.
*----------------------------------------------------------------------*
* OUTPUT: WORKFORCE ANALYTICS
*----------------------------------------------------------------------*
DATA lv_m_cnt  TYPE i VALUE 0.
DATA lv_f_cnt  TYPE i VALUE 0.
DATA lv_m_pay  TYPE i VALUE 0.
DATA lv_f_pay  TYPE i VALUE 0.
DATA lv_gs_cnt TYPE i VALUE 0.
DATA lv_ga_cnt TYPE i VALUE 0.
DATA lv_gb_cnt TYPE i VALUE 0.
DATA lv_gc_cnt TYPE i VALUE 0.
DATA lv_gs_pay TYPE i VALUE 0.
DATA lv_ga_pay TYPE i VALUE 0.
```

```
DATA lv_gb_pay TYPE i VALUE 0.
DATA lv_gc_pay TYPE i VALUE 0.
DATA lv_max_s  TYPE i VALUE 0.
DATA lv_min_s  TYPE i VALUE 9999999.
DATA lv_emp_total TYPE i VALUE 0.
DATA lv_pay_total TYPE i VALUE 0.
LOOP AT lt_emp INTO wa_emp.
  ADD 1 TO lv_emp_total.
  ADD wa_emp-basic TO lv_pay_total.
  IF wa_emp-basic > lv_max_s. lv_max_s = wa_emp-basic. ENDIF.
  IF wa_emp-basic < lv_min_s. lv_min_s = wa_emp-basic. ENDIF.
  IF wa_emp-gender = 'M'.
    ADD 1 TO lv_m_cnt. ADD wa_emp-basic TO lv_m_pay.
  ELSE.
    ADD 1 TO lv_f_cnt. ADD wa_emp-basic TO lv_f_pay.
  ENDIF.
  CASE wa_emp-grade.
    WHEN 'S'. ADD 1 TO lv_gs_cnt. ADD wa_emp-basic TO lv_gs_pay.
    WHEN 'A'. ADD 1 TO lv_ga_cnt. ADD wa_emp-basic TO lv_ga_pay.
    WHEN 'B'. ADD 1 TO lv_gb_cnt. ADD wa_emp-basic TO lv_gb_pay.
    WHEN 'C'. ADD 1 TO lv_gc_cnt. ADD wa_emp-basic TO lv_gc_pay.
  ENDCASE.
ENDLOOP.
WRITE / ' '.
WRITE / '--- WORKFORCE ANALYTICS ---'.
WRITE / 'Total Headcount :'.
WRITE / lv_emp_total.
WRITE / 'Male :'.
WRITE / lv_m_cnt.
WRITE / 'Female :'.
WRITE / lv_f_cnt.
WRITE / 'Monthly Payroll Outgo :'.
WRITE / lv_pay_total.
WRITE / 'Average Basic :'.
WRITE / lv_pay_total / lv_emp_total.
WRITE / 'Highest Basic :'.
```

```
WRITE / lv_max_s.
WRITE / 'Lowest Basic :'.
WRITE / lv_min_s.
WRITE / 'Male Avg Basic :'.
WRITE / lv_m_pay / lv_m_cnt.
WRITE / 'Female Avg Basic :'.
WRITE / lv_f_pay / lv_f_cnt.
WRITE / 'Grade S - Count / Payroll :'.
WRITE / lv_gs_cnt.
WRITE / lv_gs_pay.
WRITE / 'Grade A - Count / Payroll :'.
WRITE / lv_ga_cnt.
WRITE / lv_ga_pay.
WRITE / 'Grade B - Count / Payroll :'.
WRITE / lv_gb_cnt.
WRITE / lv_gb_pay.
WRITE / 'Grade C - Count / Payroll :'.
WRITE / lv_gc_cnt.
WRITE / lv_gc_pay.
*----------------------------------------------------------------------*
* OUTPUT: TOP 5 EARNERS (post-increment)
*----------------------------------------------------------------------*
DATA lt_seen  TYPE TABLE OF ty_emp.
DATA wa_seen  TYPE ty_emp.
DATA wa_best  TYPE ty_emp.
DATA lv_sfnd  TYPE i.
WRITE / ' '.
WRITE / '--- TOP 5 EARNERS ---'.
DO 5 TIMES.
  lv_max_s = 0.
  LOOP AT lt_emp INTO wa_emp.
    lv_sfnd = 0.
    LOOP AT lt_seen INTO wa_seen.
      IF wa_seen-empno = wa_emp-empno. lv_sfnd = 1. ENDIF.
    ENDLOOP.
    IF lv_sfnd = 0 AND wa_emp-basic > lv_max_s.
```

```
      lv_max_s = wa_emp-basic.
      wa_best  = wa_emp.
    ENDIF.
  ENDLOOP.
  APPEND wa_best TO lt_seen.
  WRITE / sy-index.
  WRITE / wa_best-ename.
  WRITE / wa_best-dept.
  WRITE / wa_best-grade.
  WRITE / wa_best-basic.
ENDDO.
*----------------------------------------------------------------------*
* OUTPUT: DATA VALIDATION
*----------------------------------------------------------------------*
DATA lv_err  TYPE i VALUE 0.
DATA lv_warn TYPE i VALUE 0.
DATA lv_ok   TYPE i VALUE 0.
DATA lv_vmsg TYPE string.
WRITE / ' '.
WRITE / '--- DATA VALIDATION ---'.
LOOP AT lt_emp INTO wa_emp.
  IF wa_emp-basic <= 0.
    CONCATENATE wa_emp-empno ': zero or negative basic' INTO lv_vmsg.
    WRITE / lv_vmsg.
    ADD 1 TO lv_err.
  ELSE.
    ADD 1 TO lv_ok.
  ENDIF.
  IF wa_emp-grade <> 'S' AND wa_emp-grade <> 'A' AND wa_emp-grade <> 'B'
AND wa_emp-grade <> 'C'.
    CONCATENATE wa_emp-empno ': invalid grade' INTO lv_vmsg.
    WRITE / lv_vmsg.
    ADD 1 TO lv_err.
  ENDIF.
  IF wa_emp-grade = 'A' AND wa_emp-basic < 50000.
    CONCATENATE wa_emp-empno ': Grade-A with basic below band' INTO
```

```
lv_vmsg.
    WRITE / lv_vmsg.
    ADD 1 TO lv_warn.
  ENDIF.
ENDLOOP.
WRITE / 'Validated :'.
WRITE / lv_ok.
WRITE / 'Warnings :'.
WRITE / lv_warn.
WRITE / 'Errors :'.
WRITE / lv_err.
IF lv_err = 0.
  WRITE / 'Status: PASS'.
ELSE.
  WRITE / 'Status: FAIL'.
ENDIF.
*-----------------------------------------------------------------------*
* OUTPUT: SYSTEM SUMMARY
*-----------------------------------------------------------------------*
DATA lv_ps_cnt TYPE i.
DATA lv_ot_cnt TYPE i.
DATA lv_lv_cnt TYPE i.
DATA lv_yr_cnt TYPE i.
DESCRIBE TABLE lt_emp     LINES lv_emp_total.
DESCRIBE TABLE lt_payslip LINES lv_ps_cnt.
DESCRIBE TABLE lt_ot      LINES lv_ot_cnt.
DESCRIBE TABLE lt_leave   LINES lv_lv_cnt.
DESCRIBE TABLE lt_yreg    LINES lv_yr_cnt.
WRITE / ' '.
WRITE /
'================================================================'.
WRITE / 'SYSTEM RUN COMPLETE - FY 2024-25'.
WRITE / 'Employees :'.
WRITE / lv_emp_total.
WRITE / 'Payslip Records (Jan 2024) :'.
WRITE / lv_ps_cnt.
```

```
WRITE / 'OT Records :'.
WRITE / lv_ot_cnt.
WRITE / 'Leave Records :'.
WRITE / lv_lv_cnt.
WRITE / 'Yearly Register Entries :'.
WRITE / lv_yr_cnt.
WRITE / 'Net Annual Disbursement (INR) :'.
WRITE / lv_yr_net.
WRITE /
'================================================================'.
sia*
"""))
```

11. Results

The system successfully generates:

- Automated payroll calculations
- Annual earnings summaries
- Department payroll variance analysis
- Overtime payment summaries
- Workforce statistics
- Employee payslips

All validation checks pass successfully, ensuring data consistency.

12. Advantages of the System

- Eliminates manual payroll errors
- Ensures statutory compliance
- Improves reporting accuracy
- Provides workforce insights
- Enhances operational efficiency

13. Limitations

- The system processes a single fiscal year simulation.
- It does not connect to external payroll databases.
- Salary payments are not integrated with banking systems.

14. Future Enhancements

The system can be enhanced by:

- Adding database persistence
- Integrating with SAP HR modules
- Creating graphical dashboards
- Implementing web-based payroll portals
- Adding real-time tax updates

15. Conclusion

The Employee Management & Payroll System demonstrates the practical implementation of enterprise payroll automation using structured programming techniques. The system integrates employee management, payroll processing, overtime tracking, leave accounting, and workforce analytics into a unified solution.

By automating salary calculations and generating comprehensive reports, the system improves efficiency, accuracy, and transparency in payroll operations. The modular design ensures scalability and allows future enhancements such as database integration and advanced analytics.

This project illustrates how enterprise payroll systems can be designed using structured data models, algorithmic calculations, and modular program architecture.