Packages

--Scenario 1

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddNewCustomer (

p\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

);

PROCEDURE UpdateCustomerDetails (

p\_id IN NUMBER,

p\_name IN VARCHAR2

);

FUNCTION GetCustomerBalance (

p\_id IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer (

p\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

) IS

BEGIN

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (p\_id, p\_name, p\_dob, p\_balance, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Customer added: ' || p\_name);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Customer ID already exists.');

END;

PROCEDURE UpdateCustomerDetails (

p\_id IN NUMBER,

p\_name IN VARCHAR2

) IS

BEGIN

UPDATE Customers

SET Name = p\_name, LastModified = SYSDATE

WHERE CustomerID = p\_id;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Customer not found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer updated.');

END IF;

END;

FUNCTION GetCustomerBalance (

p\_id IN NUMBER

) RETURN NUMBER IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Customers WHERE CustomerID = p\_id;

RETURN v\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN -1;

END;

END CustomerManagement;

/

--test add new customers

BEGIN

CustomerManagement.AddNewCustomer(50, 'Mohan Raj', TO\_DATE('1996-01-01', 'YYYY-MM-DD'), 5000);

END;

/

--test update customer

BEGIN

CustomerManagement.UpdateCustomerDetails(50, 'Mohan Raj Kumar');

END;

/

--test get balance

DECLARE

v\_bal NUMBER;

BEGIN

v\_bal := CustomerManagement.GetCustomerBalance(50);

DBMS\_OUTPUT.PUT\_LINE('Balance: ₹' || v\_bal);

END;

/

SELECT \* FRom CUSTOMERS;

--Scenario 2

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee (

p\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_position IN VARCHAR2,

p\_salary IN NUMBER,

p\_dept IN VARCHAR2,

p\_hire\_date IN DATE

);

PROCEDURE UpdateEmployeeDetails (

p\_id IN NUMBER,

p\_name IN VARCHAR2

);

FUNCTION CalculateAnnualSalary (

p\_id IN NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee (

p\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_position IN VARCHAR2,

p\_salary IN NUMBER,

p\_dept IN VARCHAR2,

p\_hire\_date IN DATE

) IS

BEGIN

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (p\_id, p\_name, p\_position, p\_salary, p\_dept, p\_hire\_date);

DBMS\_OUTPUT.PUT\_LINE('Employee hired: ' || p\_name);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Employee ID already exists.');

END;

PROCEDURE UpdateEmployeeDetails (

p\_id IN NUMBER,

p\_name IN VARCHAR2

) IS

BEGIN

UPDATE Employees

SET Name = p\_name

WHERE EmployeeID = p\_id;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Employee not found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Employee details updated.');

END IF;

END;

FUNCTION CalculateAnnualSalary (

p\_id IN NUMBER

) RETURN NUMBER IS

v\_salary NUMBER;

BEGIN

SELECT Salary INTO v\_salary FROM Employees WHERE EmployeeID = p\_id;

RETURN v\_salary \* 12;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN -1;

END;

END EmployeeManagement;

/

--hire new employee

BEGIN

EmployeeManagement.HireEmployee(

10, 'Suresh Kumar', 'Analyst', 45000, 'Finance', TO\_DATE('2022-06-01', 'YYYY-MM-DD')

);

END;

/

--update Employee

BEGIN

EmployeeManagement.UpdateEmployeeDetails(10, 'Suresh K.');

END;

/

--claculate Annual Salary

DECLARE

v\_annual NUMBER;

BEGIN

v\_annual := EmployeeManagement.CalculateAnnualSalary(10);

DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ₹' || v\_annual);

END;

/

SELECT \* FROM Employees;

-- Scenario 3

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount (

p\_account\_id IN NUMBER,

p\_customer\_id IN NUMBER,

p\_account\_type IN VARCHAR2,

p\_balance IN NUMBER

);

PROCEDURE CloseAccount (

p\_account\_id IN NUMBER

);

FUNCTION GetTotalBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount (

p\_account\_id IN NUMBER,

p\_customer\_id IN NUMBER,

p\_account\_type IN VARCHAR2,

p\_balance IN NUMBER

) IS

BEGIN

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (p\_account\_id, p\_customer\_id, p\_account\_type, p\_balance, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Account opened: ID ' || p\_account\_id);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Account ID already exists.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

PROCEDURE CloseAccount (

p\_account\_id IN NUMBER

) IS

BEGIN

DELETE FROM Accounts WHERE AccountID = p\_account\_id;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Account not found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Account closed: ID ' || p\_account\_id);

END IF;

END;

FUNCTION GetTotalBalance (

p\_customer\_id IN NUMBER

) RETURN NUMBER IS

v\_total NUMBER := 0;

BEGIN

SELECT SUM(Balance)

INTO v\_total

FROM Accounts

WHERE CustomerID = p\_customer\_id;

RETURN NVL(v\_total, 0);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

END;

END AccountOperations;

/

BEGIN

AccountOperations.OpenAccount(100, 1, 'Savings', 5000);

END;

/

BEGIN

AccountOperations.CloseAccount(100);

END;

/

DECLARE

v\_total NUMBER;

BEGIN

v\_total := AccountOperations.GetTotalBalance(1);

DBMS\_OUTPUT.PUT\_LINE('Total Balance for Customer 1: ₹' || v\_total);

END;

/













