Scenario 1: CustomerManagement Package

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddNewCustomer(

p\_CustomerID IN Customers.CustomerID%TYPE,

p\_Name IN Customers.Name%TYPE,

p\_DOB IN Customers.DOB%TYPE,

p\_Balance IN Customers.Balance%TYPE

);

PROCEDURE UpdateCustomerDetails(

p\_CustomerID IN Customers.CustomerID%TYPE,

p\_Name IN Customers.Name%TYPE,

p\_DOB IN Customers.DOB%TYPE,

p\_Balance IN Customers.Balance%TYPE

);

FUNCTION GetCustomerBalance(

p\_CustomerID IN Customers.CustomerID%TYPE

)

RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddNewCustomer(

p\_CustomerID IN Customers.CustomerID%TYPE,

p\_Name IN Customers.Name%TYPE,

p\_DOB IN Customers.DOB%TYPE,

p\_Balance IN Customers.Balance%TYPE

) IS

BEGIN

BEGIN

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

VALUES (p\_CustomerID, p\_Name, p\_DOB, p\_Balance, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || p\_CustomerID || ' already exists');

END;

END AddNewCustomer;

PROCEDURE UpdateCustomerDetails(

p\_CustomerID IN Customers.CustomerID%TYPE,

p\_Name IN Customers.Name%TYPE,

p\_DOB IN Customers.DOB%TYPE,

p\_Balance IN Customers.Balance%TYPE

) IS

BEGIN

UPDATE Customers

SET Name = p\_Name, DOB = p\_DOB, Balance = p\_Balance, LastModified = SYSDATE

WHERE CustomerID = p\_CustomerID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || p\_CustomerID || ' not found');

END UpdateCustomerDetails;

FUNCTION GetCustomerBalance(

p\_CustomerID IN Customers.CustomerID%TYPE

)

RETURN NUMBER IS

v\_Balance NUMBER;

BEGIN

SELECT Balance INTO v\_Balance FROM Customers WHERE CustomerID = p\_CustomerID;

RETURN v\_Balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Customer with ID ' || p\_CustomerID || ' not found');

RETURN NULL;

END GetCustomerBalance;

END CustomerManagement;

/

Scenario 2: EmployeeManagement Package

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireNewEmployee(

p\_EmployeeID IN Employees.EmployeeID%TYPE,

p\_Name IN Employees.Name%TYPE,

p\_Position IN Employees.Position%TYPE,

p\_Salary IN Employees.Salary%TYPE,

p\_Department IN Employees.Department%TYPE,

p\_HireDate IN Employees.HireDate%TYPE

);

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID IN Employees.EmployeeID%TYPE,

p\_Name IN Employees.Name%TYPE,

p\_Position IN Employees.Position%TYPE,

p\_Salary IN Employees.Salary%TYPE,

p\_Department IN Employees.Department%TYPE,

p\_HireDate IN Employees.HireDate%TYPE

);

FUNCTION CalculateAnnualSalary(

p\_EmployeeID IN Employees.EmployeeID%TYPE

)

RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireNewEmployee(

p\_EmployeeID IN Employees.EmployeeID%TYPE,

p\_Name IN Employees.Name%TYPE,

p\_Position IN Employees.Position%TYPE,

p\_Salary IN Employees.Salary%TYPE,

p\_Department IN Employees.Department%TYPE,

p\_HireDate IN Employees.HireDate%TYPE

) IS

BEGIN

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

VALUES (p\_EmployeeID, p\_Name, p\_Position, p\_Salary, p\_Department, p\_HireDate);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || p\_EmployeeID || ' already exists');

END HireNewEmployee;

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID IN Employees.EmployeeID%TYPE,

p\_Name IN Employees.Name%TYPE,

p\_Position IN Employees.Position%TYPE,

p\_Salary IN Employees.Salary%TYPE,

p\_Department IN Employees.Department%TYPE,

p\_HireDate IN Employees.HireDate%TYPE

) IS

BEGIN

UPDATE Employees

SET Name = p\_Name, Position = p\_Position, Salary = p\_Salary, Department = p\_Department, HireDate = p\_HireDate

WHERE EmployeeID = p\_EmployeeID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || p\_EmployeeID || ' not found');

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary(

p\_EmployeeID IN Employees.EmployeeID%TYPE

)

RETURN NUMBER IS

v\_Salary Employees.Salary%TYPE;

BEGIN

SELECT Salary INTO v\_Salary FROM Employees WHERE EmployeeID = p\_EmployeeID;

RETURN v\_Salary \* 12; -- Assuming monthly salary

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Employee with ID ' || p\_EmployeeID || ' not found');

RETURN NULL;

END CalculateAnnualSalary;

END EmployeeManagement;

/

Scenario 3: AccountOperations Package

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenNewAccount(

p\_AccountID IN Accounts.AccountID%TYPE,

p\_CustomerID IN Accounts.CustomerID%TYPE,

p\_AccountType IN Accounts.AccountType%TYPE,

p\_Balance IN Accounts.Balance%TYPE

);

PROCEDURE CloseAccount(

p\_AccountID IN Accounts.AccountID%TYPE

);

FUNCTION GetTotalBalance(

p\_CustomerID IN Accounts.CustomerID%TYPE

)

RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenNewAccount(

p\_AccountID IN Accounts.AccountID%TYPE,

p\_CustomerID IN Accounts.CustomerID%TYPE,

p\_AccountType IN Accounts.AccountType%TYPE,

p\_Balance IN Accounts.Balance%TYPE

) IS

BEGIN

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

VALUES (p\_AccountID, p\_CustomerID, p\_AccountType, p\_Balance, SYSDATE);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account with ID ' || p\_AccountID || ' already exists');

END OpenNewAccount;

PROCEDURE CloseAccount(

p\_AccountID IN Accounts.AccountID%TYPE

) IS

BEGIN

DELETE FROM Accounts WHERE AccountID = p\_AccountID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Account with ID ' || p\_AccountID || ' not found');

END CloseAccount;

FUNCTION GetTotalBalance(

p\_CustomerID IN Accounts.CustomerID%TYPE

)

RETURN NUMBER IS

v\_TotalBalance NUMBER := 0;

BEGIN

FOR r\_Account IN (SELECT Balance FROM Accounts WHERE CustomerID = p\_CustomerID) LOOP

v\_TotalBalance := v\_TotalBalance + r\_Account.Balance;

END LOOP;

RETURN v\_TotalBalance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No accounts found for Customer ID ' || p\_CustomerID);

RETURN NULL;

END GetTotalBalance;

END AccountOperations;

/