**Exercise 7: Packages**

**Scenario 1:** Group all customer-related procedures and functions into a package.

**CODE:**

**Package specification**

CREATE OR REPLACE PACKAGE CustomerManagement AS

PROCEDURE AddCustomer (

new\_customer\_id IN NUMBER,

customer\_name IN VARCHAR2,

customer\_address IN VARCHAR2

);

PROCEDURE UpdateCustomer (

customer\_id IN NUMBER,

new\_name IN VARCHAR2,

new\_address IN VARCHAR2

);

FUNCTION GetCustomerBalance (

customer\_id IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

PROCEDURE AddCustomer (

new\_customer\_id IN NUMBER,

customer\_name IN VARCHAR2,

customer\_address IN VARCHAR2

) IS

new\_customer CUSTOMERS%ROWTYPE;

BEGIN

INSERT INTO CUSTOMERS (CUSTOMER\_ID, NAME, ADDRESS)

VALUES (new\_customer\_id, customer\_name, customer\_address);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Customer ID already exists.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error adding customer: ' || SQLERRM);

END AddCustomer;

PROCEDURE UpdateCustomer (

customer\_id IN NUMBER,

new\_name IN VARCHAR2,

new\_address IN VARCHAR2

) IS

updated\_customer CUSTOMERS%ROWTYPE;

BEGIN

UPDATE CUSTOMERS

SET NAME = new\_name,

ADDRESS = new\_address

WHERE CUSTOMER\_ID = customer\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Customer ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Error updating customer: ' || SQLERRM);

END UpdateCustomer;

FUNCTION GetCustomerBalance (

customer\_id IN NUMBER

) RETURN NUMBER IS

total\_balance NUMBER;

BEGIN

SELECT SUM(BALANCE) INTO total\_balance

FROM ACCOUNTS

WHERE CUSTOMER\_ID = customer\_id;

RETURN total\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Error retrieving customer balance: ' || SQLERRM);

END GetCustomerBalance;

END CustomerManagement;

/

**Scenario 2:** Create a package to manage employee data.

**CODE:**

**Package Specification**

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee (

new\_employee\_id IN NUMBER,

employee\_name IN VARCHAR2,

employee\_salary IN NUMBER,

employee\_department IN VARCHAR2

);

PROCEDURE UpdateEmployee (

employee\_id IN NUMBER,

new\_name IN VARCHAR2,

new\_salary IN NUMBER,

new\_department IN VARCHAR2

);

FUNCTION CalculateAnnualSalary (

employee\_id IN NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee (

new\_employee\_id IN NUMBER,

employee\_name IN VARCHAR2,

employee\_salary IN NUMBER,

employee\_department IN VARCHAR2

) IS

BEGIN

INSERT INTO EMPLOYEES (EMPLOYEE\_ID, NAME, SALARY, DEPARTMENT)

VALUES (new\_employee\_id, employee\_name, employee\_salary, employee\_department);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Employee ID already exists.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error hiring employee: ' || SQLERRM);

END HireEmployee;

PROCEDURE UpdateEmployee (

employee\_id IN NUMBER,

new\_name IN VARCHAR2,

new\_salary IN NUMBER,

new\_department IN VARCHAR2

) IS

BEGIN

UPDATE EMPLOYEES

SET NAME = new\_name,

SALARY = new\_salary,

DEPARTMENT = new\_department

WHERE EMPLOYEE\_ID = employee\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Employee ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Error updating employee: ' || SQLERRM);

END UpdateEmployee;

FUNCTION CalculateAnnualSalary (

employee\_id IN NUMBER

) RETURN NUMBER IS

annual\_salary NUMBER;

BEGIN

SELECT SALARY INTO annual\_salary

FROM EMPLOYEES

WHERE EMPLOYEE\_ID = employee\_id;

RETURN annual\_salary \* 12; -- Assuming monthly salary

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Error calculating annual salary: ' || SQLERRM);

END CalculateAnnualSalary;

END EmployeeManagement;

/

**Scenario 3:** Group all account-related operations into a package.

**Package specification**

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount (

new\_account\_id IN NUMBER,

customer\_id IN NUMBER,

initial\_balance IN NUMBER

);

PROCEDURE CloseAccount (

account\_id IN NUMBER

);

FUNCTION GetTotalBalance (

customer\_id IN NUMBER

) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount (

new\_account\_id IN NUMBER,

customer\_id IN NUMBER,

initial\_balance IN NUMBER

) IS

BEGIN

INSERT INTO ACCOUNTS (ACCOUNT\_ID, CUSTOMER\_ID, BALANCE)

VALUES (new\_account\_id, customer\_id, initial\_balance);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Account ID already exists.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Error opening account: ' || SQLERRM);

END OpenAccount;

PROCEDURE CloseAccount (

account\_id IN NUMBER

) IS

BEGIN

DELETE FROM ACCOUNTS

WHERE ACCOUNT\_ID = account\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Account ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Error closing account: ' || SQLERRM);

END CloseAccount;

FUNCTION GetTotalBalance (

customer\_id IN NUMBER

) RETURN NUMBER IS

customer\_balance NUMBER;

BEGIN

SELECT SUM(BALANCE) INTO customer\_balance

FROM ACCOUNTS

WHERE CUSTOMER\_ID = customer\_id;

RETURN customer\_balance;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Error retrieving total balance: ' || SQLERRM);

END GetTotalBalance;

END AccountOperations;

/