Exercise 7: Packages

Scenario 1:

```
CREATE OR REPLACE PACKAGE BODY CustomerManagement AS
  PROCEDURE AddNewCustomer (
    p_customer_id IN NUMBER,
    p_name IN VARCHAR2,
    p_age IN NUMBER,
    p_balance IN NUMBER
  ) IS
  BEGIN
    INSERT INTO customers (customer_id, name, age, balance)
    VALUES (p_customer_id, p_name, p_age, p_balance);
    COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error: Unable to add new customer.');
  END AddNewCustomer;

  PROCEDURE UpdateCustomerDetails (
    p_customer_id IN NUMBER,
    p_name IN VARCHAR2,
    p_age IN NUMBER
  ) IS
  BEGIN
    UPDATE customers
    SET name = p_name,
      age = p_age
    WHERE customer_id = p_customer_id;
    COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error: Unable to update customer details.');
  END UpdateCustomerDetails;

  FUNCTION GetCustomerBalance (
    p_customer_id IN NUMBER
  ) RETURN NUMBER IS
    v_balance NUMBER;
  BEGIN
    SELECT balance INTO v_balance
    FROM customers
    WHERE customer_id = p_customer_id;
    RETURN v_balance;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
```

```
        RETURN NULL;
      WHEN OTHERS THEN
        RETURN NULL;
    END GetCustomerBalance;
END CustomerManagement;
/
```

Scenario 2:

```
CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

  PROCEDURE HireEmployee (

    p_employee_id IN NUMBER,

    p_name IN VARCHAR2,

    p_department_id IN NUMBER,

    p_salary IN NUMBER

  ) IS

  BEGIN

    INSERT INTO employees (employee_id, name, department_id, salary)

    VALUES (p_employee_id, p_name, p_department_id, p_salary);

    COMMIT;

  EXCEPTION

    WHEN OTHERS THEN

      ROLLBACK;

      DBMS_OUTPUT.PUT_LINE('Error: Unable to hire employee.');

  END HireEmployee;


  PROCEDURE UpdateEmployeeDetails (

    p_employee_id IN NUMBER,

    p_name IN VARCHAR2,

    p_department_id IN NUMBER,

    p_salary IN NUMBER

  ) IS

  BEGIN

    UPDATE employees

    SET name = p_name,
        department_id = p_department_id,
        salary = p_salary
      WHERE employee_id = p_employee_id;
      COMMIT;
   EXCEPTION
      WHEN OTHERS THEN
         ROLLBACK;
         DBMS_OUTPUT.PUT_LINE('Error: Unable to update employee details.');
   END UpdateEmployeeDetails;

   FUNCTION CalculateAnnualSalary (
      p_employee_id IN NUMBER
   ) RETURN NUMBER IS
      v_salary NUMBER;
      v_annual_salary NUMBER;
   BEGIN
      SELECT salary INTO v_salary
      FROM employees
      WHERE employee_id = p_employee_id;

      v_annual_salary := v_salary * 12;
      RETURN v_annual_salary;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RETURN NULL;
      WHEN OTHERS THEN
         RETURN NULL;
   END CalculateAnnualSalary;
END EmployeeManagement;
/
```

Scenario 3:

```
CREATE OR REPLACE PACKAGE BODY AccountOperations AS
  PROCEDURE OpenNewAccount (
    p_account_id IN NUMBER,
    p_customer_id IN NUMBER,
    p_account_type IN VARCHAR2,
    p_balance IN NUMBER
  ) IS
  BEGIN
    INSERT INTO accounts (account_id, customer_id, account_type, balance)
    VALUES (p_account_id, p_customer_id, p_account_type, p_balance);
    COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error: Unable to open new account.');
  END OpenNewAccount;

  PROCEDURE CloseAccount (
    p_account_id IN NUMBER
  ) IS
  BEGIN
    DELETE FROM accounts
    WHERE account_id = p_account_id;
    COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error: Unable to close account.');
  END CloseAccount;

  FUNCTION GetTotalCustomerBalance (
        p_customer_id IN NUMBER
    ) RETURN NUMBER IS
        v_total_balance NUMBER;
    BEGIN
        SELECT SUM(balance) INTO v_total_balance
        FROM accounts
        WHERE customer_id = p_customer_id;

        RETURN v_total_balance;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN 0;
        WHEN OTHERS THEN
            RETURN 0;
    END GetTotalCustomerBalance;
END AccountOperations;
/
```