

```
SQL> CREATE OR REPLACE PACKAGE CustomerManagement AS
  2     PROCEDURE AddCustomer(
  3         p_customer_id NUMBER,
  4         p_name VARCHAR2,
  5         p_dob DATE,
  6         p_balance NUMBER
  7     );
  8
  9     PROCEDURE UpdateCustomerDetails(
10         p_customer_id NUMBER,
11         p_name VARCHAR2,
12         p_dob DATE
13     );
14
15     FUNCTION GetCustomerBalance(
16         p_customer_id NUMBER
17     ) RETURN NUMBER;
18 END CustomerManagement;
19 /
```

Package created.

```

SQL> CREATE OR REPLACE PACKAGE BODY CustomerManagement AS
2
3     PROCEDURE AddCustomer(
4         p_customer_id NUMBER,
5         p_name VARCHAR2,
6         p_dob DATE,
7         p_balance NUMBER
8     ) IS
9     BEGIN
10         INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
11         VALUES (p_customer_id, p_name, p_dob, p_balance, SYSDATE);
12
13         DBMS_OUTPUT.PUT_LINE('Customer added: ' || p_name);
14     EXCEPTION
15         WHEN DUP_VAL_ON_INDEX THEN
16             DBMS_OUTPUT.PUT_LINE('Customer ID already exists.');
```

```

17     END AddCustomer;
18
19     PROCEDURE UpdateCustomerDetails(
20         p_customer_id NUMBER,
21         p_name VARCHAR2,
22         p_dob DATE
23     ) IS
24     BEGIN
25         UPDATE Customers
26         SET Name = p_name, DOB = p_dob, LastModified = SYSDATE
27         WHERE CustomerID = p_customer_id;
28
29         DBMS_OUTPUT.PUT_LINE('Customer details updated for ID: ' || p_customer_id);
30     END UpdateCustomerDetails;
31
32     FUNCTION GetCustomerBalance(
33         p_customer_id NUMBER
34     ) RETURN NUMBER IS
35         v_balance NUMBER;
36     BEGIN
37         SELECT Balance INTO v_balance FROM Customers
38         WHERE CustomerID = p_customer_id;
39
40         RETURN v_balance;
41     EXCEPTION
42         WHEN NO_DATA_FOUND THEN
43             RETURN NULL;
44     END GetCustomerBalance;
45
46 END CustomerManagement;
47 /

```

Package body created.

```
SQL> CREATE OR REPLACE PACKAGE EmployeeManagement AS
```

```
2     PROCEDURE HireEmployee(  
3         p_employee_id NUMBER,  
4         p_name VARCHAR2,  
5         p_position VARCHAR2,  
6         p_salary NUMBER,  
7         p_department VARCHAR2,  
8         p_hire_date DATE  
9     );  
10  
11     PROCEDURE UpdateEmployeeDetails(  
12         p_employee_id NUMBER,  
13         p_position VARCHAR2,  
14         p_salary NUMBER,  
15         p_department VARCHAR2  
16     );  
17  
18     FUNCTION CalculateAnnualSalary(  
19         p_employee_id NUMBER  
20     ) RETURN NUMBER;  
21 END EmployeeManagement;  
22 /
```

Package created.

```

SQL> CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS
2
3     PROCEDURE HireEmployee(
4         p_employee_id NUMBER,
5         p_name VARCHAR2,
6         p_position VARCHAR2,
7         p_salary NUMBER,
8         p_department VARCHAR2,
9         p_hire_date DATE
10    ) IS
11 BEGIN
12     INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)
13     VALUES (p_employee_id, p_name, p_position, p_salary, p_department, p_hire_date);
14
15     DBMS_OUTPUT.PUT_LINE('Employee hired: ' || p_name);
16 EXCEPTION
17     WHEN DUP_VAL_ON_INDEX THEN
18         DBMS_OUTPUT.PUT_LINE('Employee ID already exists.');
```

```

19 END HireEmployee;
20
21 PROCEDURE UpdateEmployeeDetails(
22     p_employee_id NUMBER,
23     p_position VARCHAR2,
24     p_salary NUMBER,
25     p_department VARCHAR2
26 ) IS
27 BEGIN
28     UPDATE Employees
29     SET Position = p_position, Salary = p_salary, Department = p_department
30     WHERE EmployeeID = p_employee_id;
31
32     DBMS_OUTPUT.PUT_LINE('Employee details updated for ID: ' || p_employee_id);
33 END UpdateEmployeeDetails;
34
35 FUNCTION CalculateAnnualSalary(
36     p_employee_id NUMBER
37 ) RETURN NUMBER IS
38     v_monthly_salary NUMBER;
39 BEGIN
40     SELECT Salary INTO v_monthly_salary FROM Employees
41     WHERE EmployeeID = p_employee_id;
42
43     RETURN v_monthly_salary * 12;
44 EXCEPTION
45     WHEN NO_DATA_FOUND THEN
46         RETURN NULL;
47 END CalculateAnnualSalary;
48
49 END EmployeeManagement;
50 /

```

```
SQL> CREATE OR REPLACE PACKAGE AccountOperations AS
2     PROCEDURE OpenAccount(
3         p_account_id NUMBER,
4         p_customer_id NUMBER,
5         p_account_type VARCHAR2,
6         p_balance NUMBER
7     );
8
9     PROCEDURE CloseAccount(
10        p_account_id NUMBER
11    );
12
13    FUNCTION GetTotalCustomerBalance(
14        p_customer_id NUMBER
15    ) RETURN NUMBER;
16 END AccountOperations;
17 /
```

Package created.

```
SQL> CREATE OR REPLACE PACKAGE BODY AccountOperations AS
```

```
2
3     PROCEDURE OpenAccount(
4         p_account_id NUMBER,
5         p_customer_id NUMBER,
6         p_account_type VARCHAR2,
7         p_balance NUMBER
8     ) IS
9     BEGIN
10         INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)
11         VALUES (p_account_id, p_customer_id, p_account_type, p_balance, SYSDATE);
12
13         DBMS_OUTPUT.PUT_LINE('Account opened: ' || p_account_id);
14     EXCEPTION
15         WHEN DUP_VAL_ON_INDEX THEN
16             DBMS_OUTPUT.PUT_LINE('Account ID already exists.');
```

```
17     END OpenAccount;
18
19     PROCEDURE CloseAccount(
20         p_account_id NUMBER
21     ) IS
22     BEGIN
23         DELETE FROM Accounts
24         WHERE AccountID = p_account_id;
25
26         DBMS_OUTPUT.PUT_LINE('Account closed: ' || p_account_id);
27     END CloseAccount;
28
29     FUNCTION GetTotalCustomerBalance(
30         p_customer_id NUMBER
31     ) RETURN NUMBER IS
32         v_total_balance NUMBER;
33     BEGIN
34         SELECT SUM(Balance) INTO v_total_balance
35         FROM Accounts
36         WHERE CustomerID = p_customer_id;
37
38         RETURN NUL(v_total_balance, 0);
39     END GetTotalCustomerBalance;
40
41 END AccountOperations;
42 /
```

Package body created.