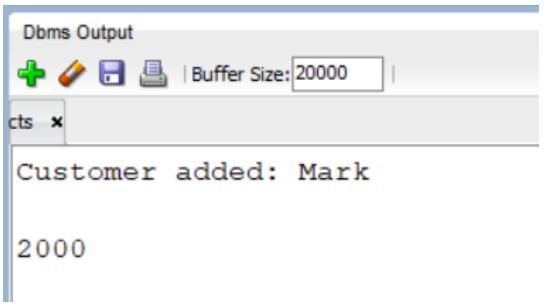
**Exercise 7: Packages**

**Scenario 1:** Group all customer-related procedures and functions into a package.
**Question:** Create a package **CustomerManagement** with procedures for adding a new customer, updating customer details, and a function to get customer balance.

**Solution:**

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PACKAGE CustomerManagement AS
   PROCEDURE AddCustomer(p_id NUMBER, p_name VARCHAR2, p_dob DATE,
p_balance NUMBER);
   PROCEDURE UpdateCustomer(p_id NUMBER, p_name VARCHAR2, p_dob DATE);
   FUNCTION GetCustomerBalance(p_id NUMBER) RETURN NUMBER;
END CustomerManagement;
/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

   PROCEDURE AddCustomer(p_id NUMBER, p_name VARCHAR2, p_dob DATE,
p_balance 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);
   END;

   PROCEDURE UpdateCustomer(p_id NUMBER, p_name VARCHAR2, p_dob DATE) IS
   BEGIN
      UPDATE Customers
      SET Name = p_name,
         DOB = p_dob,
         LastModified = SYSDATE
      WHERE CustomerID = p_id;

      DBMS_OUTPUT.PUT_LINE('Customer updated: ID ' || p_id);
   END;

   FUNCTION GetCustomerBalance(p_id 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 NULL;
  END;

END CustomerManagement;
/

EXEC CustomerManagement.AddCustomer(3, 'Mark', TO_DATE('1995-02-10',
'YYYY-MM-DD'), 2000);
EXEC DBMS_OUTPUT.PUT_LINE(CustomerManagement.GetCustomerBalance(3));
```

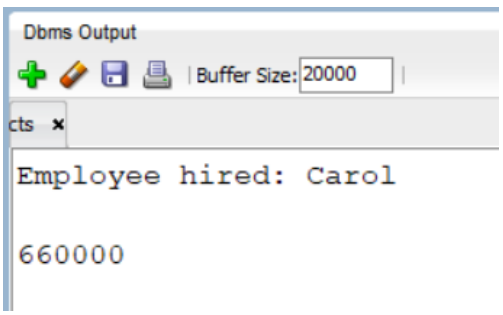



**Scenario 2:** Create a package to manage employee data.
**Question:** Write a package **EmployeeManagement** with procedures to hire new employees, update employee details, and a function to calculate annual salary.

**Solution:**

```
CREATE OR REPLACE PACKAGE EmployeeManagement AS
  PROCEDURE HireEmployee(p_id NUMBER, p_name VARCHAR2, p_position
VARCHAR2, p_salary NUMBER, p_dept VARCHAR2, p_hiredate DATE);
  PROCEDURE UpdateEmployee(p_id NUMBER, p_position VARCHAR2, p_salary
NUMBER);
  FUNCTION CalculateAnnualSalary(p_id NUMBER) RETURN NUMBER;
END EmployeeManagement;
/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

  PROCEDURE HireEmployee(p_id NUMBER, p_name VARCHAR2, p_position
VARCHAR2, p_salary NUMBER, p_dept VARCHAR2, p_hiredate DATE) IS
  BEGIN
```

```
        INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department,
HireDate)
        VALUES (p_id, p_name, p_position, p_salary, p_dept, p_hiredate);
        DBMS_OUTPUT.PUT_LINE('Employee hired: ' || p_name);
    END;

    PROCEDURE UpdateEmployee(p_id NUMBER, p_position VARCHAR2, p_salary
NUMBER) IS
    BEGIN
        UPDATE Employees
        SET Position = p_position,
            Salary = p_salary
        WHERE EmployeeID = p_id;
        DBMS_OUTPUT.PUT_LINE('Employee updated: ID ' || p_id);
    END;

    FUNCTION CalculateAnnualSalary(p_id 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 NULL;
    END;

END EmployeeManagement;
/

EXEC EmployeeManagement.HireEmployee(3, 'Carol', 'Analyst', 55000, 'Finance',
SYSDATE);
EXEC DBMS_OUTPUT.PUT_LINE(EmployeeManagement.CalculateAnnualSalary(3));
```



**Scenario 3:** Group all account-related operations into a package.
**Question:** Create a package **AccountOperations** with procedures for opening a new account, closing an account, and a function to get the total balance of a customer across all accounts.

**Solution:**

```
CREATE OR REPLACE PACKAGE AccountOperations AS
   PROCEDURE OpenAccount(p_acc_id NUMBER, p_cust_id NUMBER, p_type
VARCHAR2, p_balance NUMBER);
   PROCEDURE CloseAccount(p_acc_id NUMBER);
   FUNCTION GetTotalBalance(p_cust_id NUMBER) RETURN NUMBER;
END AccountOperations;
/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

   PROCEDURE OpenAccount(p_acc_id NUMBER, p_cust_id NUMBER, p_type
VARCHAR2, p_balance NUMBER) IS
   BEGIN
      INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance,
LastModified)
      VALUES (p_acc_id, p_cust_id, p_type, p_balance, SYSDATE);
      DBMS_OUTPUT.PUT_LINE('Account opened: ID ' || p_acc_id);
   END;

   PROCEDURE CloseAccount(p_acc_id NUMBER) IS
   BEGIN
      DELETE FROM Accounts WHERE AccountID = p_acc_id;
      DBMS_OUTPUT.PUT_LINE('Account closed: ID ' || p_acc_id);
   END;

   FUNCTION GetTotalBalance(p_cust_id NUMBER) RETURN NUMBER IS
      v_total NUMBER := 0;
   BEGIN
      SELECT NVL(SUM(Balance), 0) INTO v_total
      FROM Accounts
      WHERE CustomerID = p_cust_id;
      RETURN v_total;
   END;

END AccountOperations;
/

EXEC AccountOperations.OpenAccount(3, 3, 'Savings', 3000);
```

```
EXEC DBMS_OUTPUT.PUT_LINE('Total Balance: ' ||
AccountOperations.GetTotalBalance(3));
```