Q1.

CREATE SEQUENCE CustomerID\_seq

START WITH 4 -- Assuming you already have customers with IDs 1, 2, and 3

INCREMENT BY 1

NOCACHE

NOCYCLE;

/

CREATE OR REPLACE PACKAGE CustomerManagement AS

-- Procedure to add a new customer

PROCEDURE AddCustomer(

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

);

-- Procedure to update customer details

PROCEDURE UpdateCustomer(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

);

-- Function to get customer balance

FUNCTION GetCustomerBalance(

p\_CustomerID IN NUMBER

) RETURN NUMBER;

END CustomerManagement;

/

CREATE OR REPLACE PACKAGE BODY CustomerManagement AS

-- Implementation of AddCustomer procedure

PROCEDURE AddCustomer(

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

) IS

v\_CustomerID NUMBER;

BEGIN

-- Generate a new unique CustomerID

SELECT CustomerID\_seq.NEXTVAL INTO v\_CustomerID FROM DUAL;

-- Insert the new customer into the Customers table

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

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

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE\_APPLICATION\_ERROR(-20001, 'Error adding customer: ' || SQLERRM);

END AddCustomer;

-- Implementation of UpdateCustomer procedure

PROCEDURE UpdateCustomer(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

) IS

BEGIN

UPDATE Customers

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

WHERE CustomerID = p\_CustomerID;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END UpdateCustomer;

-- Implementation of GetCustomerBalance function

FUNCTION GetCustomerBalance(

p\_CustomerID IN NUMBER

) 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

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

WHEN OTHERS THEN

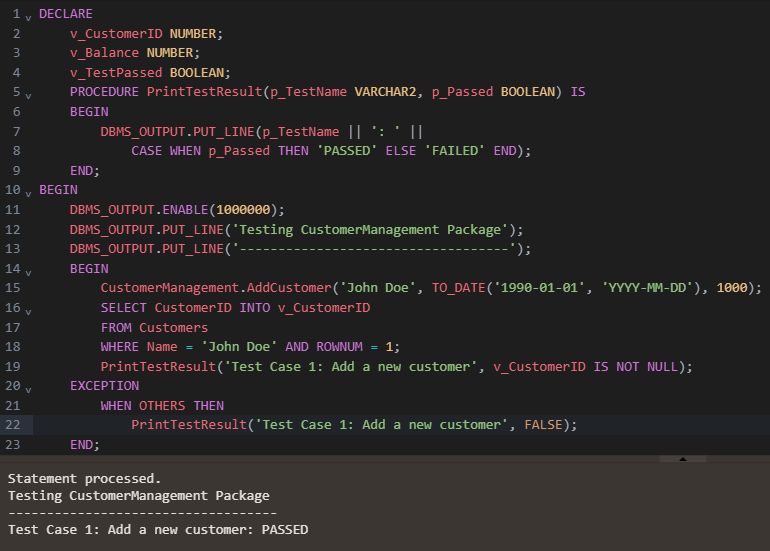
RAISE\_APPLICATION\_ERROR(-20004, 'Error retrieving balance: ' || SQLERRM);

END GetCustomerBalance;

END CustomerManagement;

/

*Output:*

**

Q2.

CREATE OR REPLACE PACKAGE EmployeeManagement AS

PROCEDURE HireEmployee(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2,

p\_Position VARCHAR2,

p\_Salary NUMBER,

p\_Department VARCHAR2,

p\_HireDate DATE

);

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2 := NULL,

p\_Position VARCHAR2 := NULL,

p\_Salary NUMBER := NULL,

p\_Department VARCHAR2 := NULL

);

FUNCTION CalculateAnnualSalary(

p\_EmployeeID NUMBER

) RETURN NUMBER;

END EmployeeManagement;

/

CREATE OR REPLACE PACKAGE BODY EmployeeManagement AS

PROCEDURE HireEmployee(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2,

p\_Position VARCHAR2,

p\_Salary NUMBER,

p\_Department VARCHAR2,

p\_HireDate DATE

) IS

BEGIN

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

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

END HireEmployee;

PROCEDURE UpdateEmployeeDetails(

p\_EmployeeID NUMBER,

p\_Name VARCHAR2 := NULL,

p\_Position VARCHAR2 := NULL,

p\_Salary NUMBER := NULL,

p\_Department VARCHAR2 := NULL

) IS

BEGIN

UPDATE Employees

SET

Name = NVL(p\_Name, Name),

Position = NVL(p\_Position, Position),

Salary = NVL(p\_Salary, Salary),

Department = NVL(p\_Department, Department)

WHERE EmployeeID = p\_EmployeeID;

END UpdateEmployeeDetails;

FUNCTION CalculateAnnualSalary(

p\_EmployeeID NUMBER

) RETURN NUMBER IS

v\_Salary NUMBER;

BEGIN

SELECT Salary INTO v\_Salary

FROM Employees

WHERE EmployeeID = p\_EmployeeID;

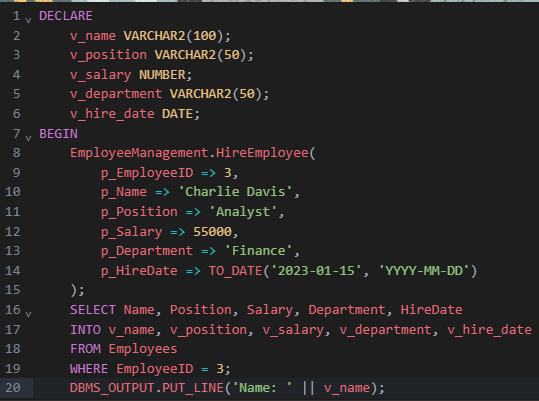
RETURN v\_Salary \* 12;

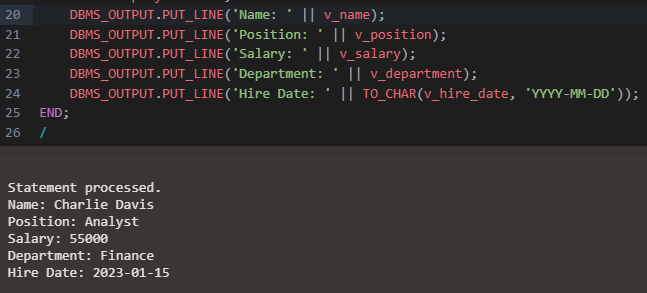
END CalculateAnnualSalary;

END EmployeeManagement;

/

*Output:*

**

**

Q3.

CREATE OR REPLACE PACKAGE AccountOperations AS

PROCEDURE OpenAccount(

p\_AccountID NUMBER,

p\_CustomerID NUMBER,

p\_AccountType VARCHAR2,

p\_Balance NUMBER

);

PROCEDURE CloseAccount(

p\_AccountID NUMBER

);

FUNCTION GetTotalBalance(

p\_CustomerID NUMBER

) RETURN NUMBER;

END AccountOperations;

/

CREATE OR REPLACE PACKAGE BODY AccountOperations AS

PROCEDURE OpenAccount(

p\_AccountID NUMBER,

p\_CustomerID NUMBER,

p\_AccountType VARCHAR2,

p\_Balance NUMBER

) IS

BEGIN

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

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

END OpenAccount;

PROCEDURE CloseAccount(

p\_AccountID NUMBER

) IS

BEGIN

DELETE FROM Accounts

WHERE AccountID = p\_AccountID;

END CloseAccount;

FUNCTION GetTotalBalance(

p\_CustomerID NUMBER

) RETURN NUMBER IS

v\_TotalBalance NUMBER;

BEGIN

SELECT SUM(Balance)

INTO v\_TotalBalance

FROM Accounts

WHERE CustomerID = p\_CustomerID;

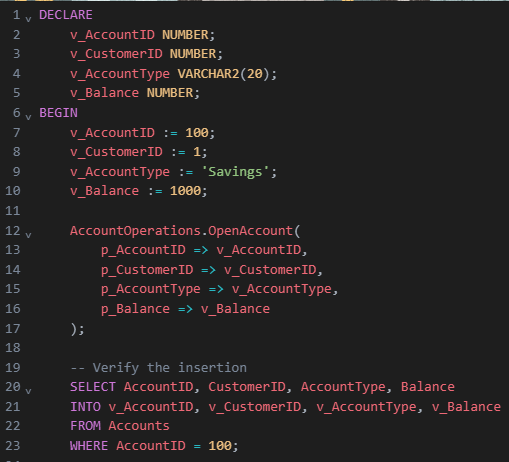
RETURN NVL(v\_TotalBalance, 0);

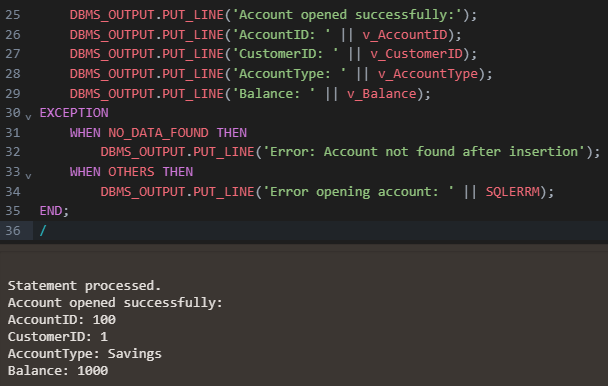
END GetTotalBalance;

END AccountOperations;

/

*Output:*

**

**