**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loans CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(50),

age NUMBER,

balance NUMBER(10,2),

isVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES customers(customer\_id),

interest\_rate NUMBER(5,3),

due\_date DATE

);

INSERT INTO customers VALUES (1,'Lohith',65,12000,'N');

INSERT INTO customers VALUES (2,'Bob',58,8000,'N');

INSERT INTO customers VALUES (3,'JLK',70,15000,'N');

INSERT INTO loans VALUES (101,1,0.075,SYSDATE+10);

INSERT INTO loans VALUES (102,2,0.065,SYSDATE+40);

INSERT INTO loans VALUES (103,3,0.080,SYSDATE+25);

COMMIT;

BEGIN

FOR r IN (SELECT customer\_id FROM customers WHERE age > 60) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 0.01

WHERE customer\_id = r.customer\_id;

END LOOP;

COMMIT;

END;

/

BEGIN

UPDATE customers

SET isVIP = 'Y'

WHERE balance > 10000;

COMMIT;

END;

/

BEGIN

FOR r IN (

SELECT l.loan\_id, c.customer\_name, l.due\_date

FROM loans l

JOIN customers c ON c.customer\_id = l.customer\_id

WHERE l.due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || r.loan\_id || ' for ' || r.customer\_name || ' is due on ' || TO\_CHAR(r.due\_date, 'DD-Mon-YYYY'));

END LOOP;

END;

/

BEGIN

FOR r IN (

SELECT l.loan\_id, c.customer\_name, c.age, l.interest\_rate

FROM loans l

JOIN customers c ON c.customer\_id = l.customer\_id

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Loan ' || r.loan\_id || ' | ' || r.customer\_name || ' | Age: ' || r.age || ' | Rate: ' || r.interest\_rate);

END LOOP;

END;

/

BEGIN

FOR r IN (

SELECT customer\_id, customer\_name, balance, isVIP FROM customers

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Customer ' || r.customer\_id || ' | ' || r.customer\_name || ' | Balance: ' || r.balance || ' | VIP: ' || r.isVIP);

END LOOP;

END;

/

**Scenario 1**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.Scenario 3**

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees CASCADE CONSTRAINTS';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(50),

account\_type VARCHAR2(20),

balance NUMBER(10,2)

);

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

department VARCHAR2(30),

salary NUMBER(10,2)

);

INSERT INTO accounts VALUES (1, 'Alice', 'Savings', 1000);

INSERT INTO accounts VALUES (2, 'Bob', 'Savings', 2000);

INSERT INTO accounts VALUES (3, 'Alice', 'Checking', 1500);

INSERT INTO accounts VALUES (4, 'Bob', 'Checking', 2500);

INSERT INTO employees VALUES (101, 'John', 'HR', 40000);

INSERT INTO employees VALUES (102, 'Jane', 'IT', 60000);

INSERT INTO employees VALUES (103, 'Mike', 'IT', 55000);

COMMIT;

/

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT account\_id, balance FROM accounts WHERE account\_type = 'Savings') LOOP

UPDATE accounts

SET balance = rec.balance + (rec.balance \* 0.01)

WHERE account\_id = rec.account\_id;

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

FOR rec IN (SELECT employee\_id, salary FROM employees WHERE department = dept\_name) LOOP

UPDATE employees

SET salary = rec.salary + (rec.salary \* bonus\_percent / 100)

WHERE employee\_id = rec.employee\_id;

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_acct IN NUMBER,

to\_acct IN NUMBER,

amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance FROM accounts WHERE account\_id = from\_acct FOR UPDATE;

IF v\_balance < amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance');

END IF;

UPDATE accounts SET balance = balance - amount WHERE account\_id = from\_acct;

UPDATE accounts SET balance = balance + amount WHERE account\_id = to\_acct;

END;

/

BEGIN

ProcessMonthlyInterest;

UpdateEmployeeBonus('IT', 10);

TransferFunds(3, 4, 500);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Final Account Balances ---');

FOR r IN (SELECT \* FROM accounts ORDER BY account\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE('ID: '||r.account\_id||', Name: '||r.customer\_name||', Type: '||r.account\_type||', Balance: '||r.balance);

END LOOP;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Final Employee Salaries ---');

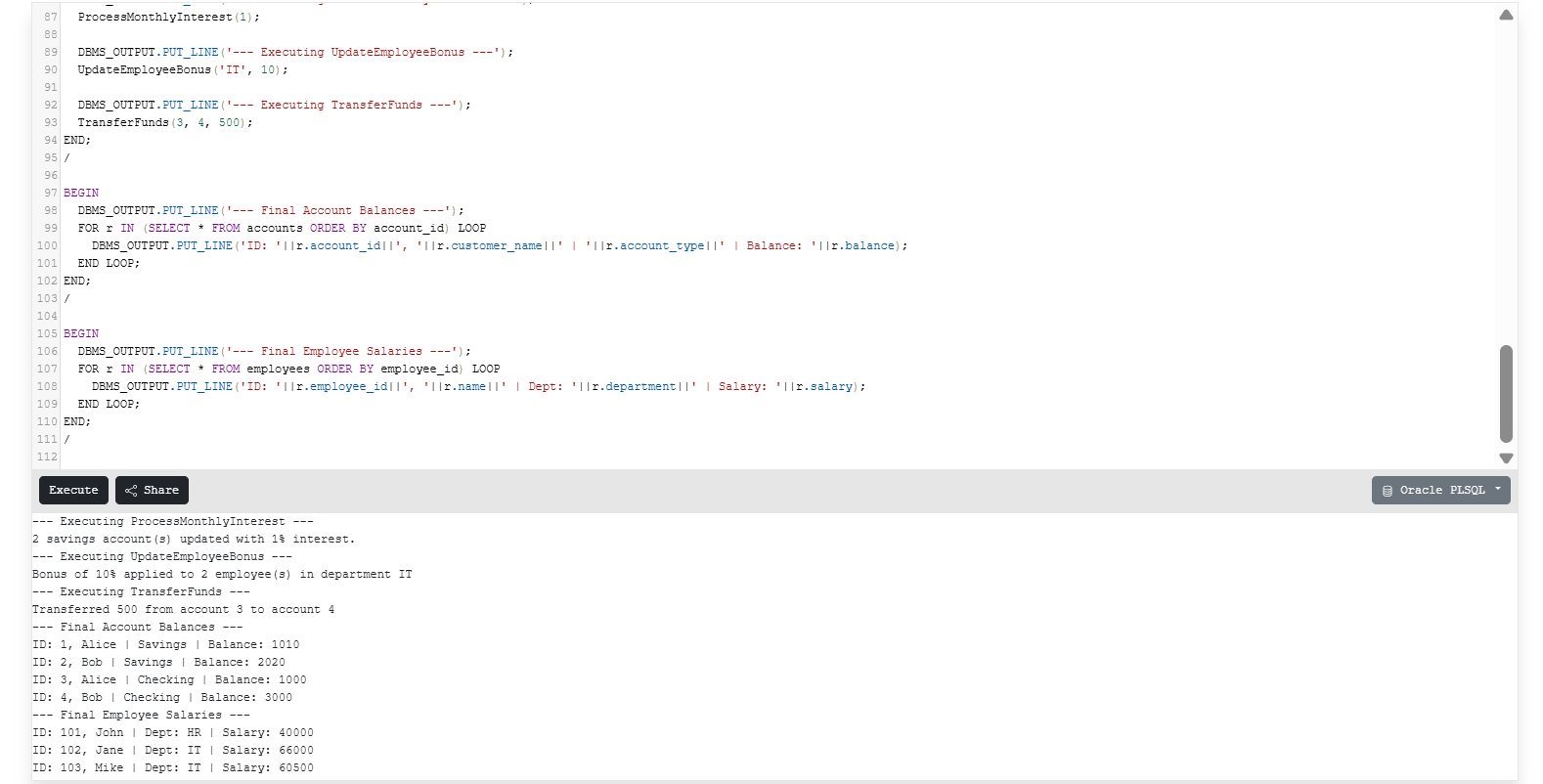
FOR r IN (SELECT \* FROM employees ORDER BY employee\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE('ID: '||r.employee\_id||', Name: '||r.name||', Dept: '||r.department||', Salary: '||r.salary);

END LOOP;

END;

/



**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.

**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.

**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.

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-05-15','YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20','YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

COMMIT;

-- Add new customer

CREATE OR REPLACE PROCEDURE AddCustomer (

p\_id NUMBER, p\_name VARCHAR2, p\_dob DATE, p\_balance NUMBER

) AS

BEGIN

INSERT INTO Customers VALUES (p\_id, p\_name, p\_dob, p\_balance, SYSDATE);

END;

/

-- Update customer details

CREATE OR REPLACE PROCEDURE UpdateCustomer (

p\_id NUMBER, p\_name VARCHAR2, p\_balance NUMBER

) AS

BEGIN

UPDATE Customers SET Name = p\_name, Balance = p\_balance, LastModified = SYSDATE WHERE CustomerID = p\_id;

END;

/

-- Get customer balance

CREATE OR REPLACE FUNCTION GetCustomerBalance (

p\_id NUMBER

) RETURN NUMBER AS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Customers WHERE CustomerID = p\_id;

RETURN v\_balance;

END;

/

-- Hire employee

CREATE OR REPLACE PROCEDURE HireEmployee (

p\_id NUMBER, p\_name VARCHAR2, p\_pos VARCHAR2, p\_sal NUMBER, p\_dept VARCHAR2, p\_date DATE

) AS

BEGIN

INSERT INTO Employees VALUES (p\_id, p\_name, p\_pos, p\_sal, p\_dept, p\_date);

END;

/

-- Update employee details

CREATE OR REPLACE PROCEDURE UpdateEmployee (

p\_id NUMBER, p\_salary NUMBER

) AS

BEGIN

UPDATE Employees SET Salary = p\_salary WHERE EmployeeID = p\_id;

END;

/

-- Get annual salary

CREATE OR REPLACE FUNCTION GetAnnualSalary (

p\_id NUMBER

) RETURN NUMBER AS

v\_salary NUMBER;

BEGIN

SELECT Salary INTO v\_salary FROM Employees WHERE EmployeeID = p\_id;

RETURN v\_salary \* 12;

END;

/

-- Open account

CREATE OR REPLACE PROCEDURE OpenAccount (

p\_id NUMBER, p\_cust\_id NUMBER, p\_type VARCHAR2, p\_balance NUMBER

) AS

BEGIN

INSERT INTO Accounts VALUES (p\_id, p\_cust\_id, p\_type, p\_balance, SYSDATE);

END;

/

-- Close account

CREATE OR REPLACE PROCEDURE CloseAccount (

p\_id NUMBER

) AS

BEGIN

DELETE FROM Accounts WHERE AccountID = p\_id;

END;

/

-- Get total balance of customer

CREATE OR REPLACE FUNCTION GetTotalBalance (

p\_cust\_id NUMBER

) RETURN NUMBER AS

v\_total NUMBER;

BEGIN

SELECT NVL(SUM(Balance), 0) INTO v\_total FROM Accounts WHERE CustomerID = p\_cust\_id;

RETURN v\_total;

END;

/

-- Run test cases

BEGIN

AddCustomer(3, 'Lohith', TO\_DATE('1995-09-09','YYYY-MM-DD'), 2000);

UpdateCustomer(3, 'Lohith Kumar', 2500);

HireEmployee(3, 'Nina Rao', 'Tester', 50000, 'QA', SYSDATE);

UpdateEmployee(3, 55000);

OpenAccount(3, 3, 'Savings', 2500);

END;

/

-- Check results

SELECT Name, Balance FROM Customers WHERE CustomerID = 3;

SELECT Name, Salary FROM Employees WHERE EmployeeID = 3;

SELECT AccountType, Balance FROM Accounts WHERE CustomerID = 3;

SELECT GetCustomerBalance(3) AS Customer\_Balance FROM dual;

SELECT GetAnnualSalary(3) AS Annual\_Salary FROM dual;

SELECT GetTotalBalance(3) AS Total\_Account\_Balance FROM dual;

A screenshot of a computer

AI-generated content may be incorrect.