**PL/SQL programming**

***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.**

BEGIN

FOR cust IN (

SELECT customer\_id

FROM customers

WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE, date\_of\_birth) / 12) > 60

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 0.01

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

END;

**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.**

BEGIN

FOR cust IN (

SELECT customer\_id

FROM customers

WHERE balance > 10000

) LOOP

UPDATE customers

SET is\_vip = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

END;

**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.**

DECLARE

CURSOR cur\_loans IS

SELECT loan\_id, customer\_id, due\_date

FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;

v\_customer\_name customers.name%TYPE;

BEGIN

FOR loan\_rec IN cur\_loans LOOP

SELECT name INTO v\_customer\_name

FROM customers

WHERE customer\_id = loan\_rec.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.loan\_id ||

' for customer ' || v\_customer\_name ||

' is due on ' || TO\_CHAR(loan\_rec.due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

***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.**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'SAVINGS';

COMMIT;

END;

**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.**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department\_id IN employees.department\_id%TYPE,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

SET salary = salary + (salary \* p\_bonus\_percent / 100)

WHERE department\_id = p\_department\_id;

COMMIT;

END;

**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.**

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_account\_id

FOR UPDATE;

IF from\_balance < amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account\_id;

COMMIT;

END;

/