**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.

**CODE:**

BEGIN

FOR rec IN (

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

FROM customers c

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

WHERE c.age > 60

)

LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE loan\_id = rec.loan\_id;

DBMS\_OUTPUT.put\_line(

'1% discount applied to Loan ' || rec.loan\_id ||

' for Customer#' || rec.customer\_id

);

END LOOP;

COMMIT;

END;

/

**OUTPUT:**

1% discount applied to Loan 9001 for Customer#101

1% discount applied to Loan 9002 for Customer#102

**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.

**CODE:**

BEGIN

FOR rec IN (

SELECT customer\_id, balance

FROM customers

WHERE balance > 10000 AND isvip <> 'Y'

)

LOOP

UPDATE customers

SET isvip = 'Y'

WHERE customer\_id = rec.customer\_id;

DBMS\_OUTPUT.put\_line(

'Customer#' || rec.customer\_id ||

' promoted to VIP (Balance: $' || rec.balance || ')'

);

END LOOP;

COMMIT;

END;

/

**OUTPUT:**

Customer#102 promoted to VIP (Balance: $12700)

**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.

**CODE:**

BEGIN

FOR rec IN (

SELECT l.loan\_id,

l.customer\_id,

l.due\_date,

c.first\_name || ' ' || c.last\_name AS cust\_name

FROM loans l

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

WHERE l.due\_date BETWEEN TRUNC(SYSDATE) AND TRUNC(SYSDATE) + 30

ORDER BY l.due\_date

)

LOOP

DBMS\_OUTPUT.put\_line(

'REMINDER --> Loan ' || rec.loan\_id ||

' for ' || rec.cust\_name ||

' (Cust#' || rec.customer\_id || ') is due on ' ||

TO\_CHAR(rec.due\_date, 'DD-MON-YYYY')

);

END LOOP;

END;

/

**OUTPUT:**

REMINDER --> Loan 9002 for Jane Smith (Cust#102) is due on 03-JUL-2025

REMINDER --> Loan 9004 for Alice Brown (Cust#104) is due on 08-JUL-2025

REMINDER --> Loan 9001 for John Doe (Cust#101) is due on 13-JUL-2025

**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.

**CODE:**

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 = balance + (balance \* 0.01)

WHERE account\_id = rec.account\_id;

DBMS\_OUTPUT.put\_line('Interest added to Account#' || rec.account\_id);

END LOOP;

COMMIT;

END;

/

**OUTPUT:**

Interest added to Account#201, New balance: 5050.00

Interest added to Account#202, New balance: 15150.00

**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.

**CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

v\_old\_salary employees.salary%TYPE;

v\_bonus\_amount NUMBER(12,2);

CURSOR c\_emps IS

SELECT employee\_id, name, salary FROM employees WHERE department = p\_department FOR UPDATE;

BEGIN

FOR emp\_rec IN c\_emps LOOP

v\_old\_salary := emp\_rec.salary;

v\_bonus\_amount := ROUND(v\_old\_salary \* p\_bonus\_percent / 100, 2);

UPDATE employees

SET salary = salary + v\_bonus\_amount

WHERE employee\_id = emp\_rec.employee\_id;

DBMS\_OUTPUT.put\_line('Employee: ' || emp\_rec.name ||

' | Old Salary: ' || TO\_CHAR(v\_old\_salary, '999999.99') ||

' | Bonus: ' || TO\_CHAR(v\_bonus\_amount, '999999.99') ||

' | New Salary: ' || TO\_CHAR(v\_old\_salary + v\_bonus\_amount, '999999.99'));

END LOOP;

COMMIT;

DBMS\_OUTPUT.put\_line('Bonus updated for Department: ' || p\_department);

END;

/

BEGIN

DBMS\_OUTPUT.put\_line('Running UpdateEmployeeBonus...');

UpdateEmployeeBonus('Sales', 10);

END;

/

**Output:**

Running UpdateEmployeeBonus...

Employee: John Doe | Old Salary: 4000.00 | Bonus: 400.00 | New Salary:

4400.00

Employee: Jane Smith | Old Salary: 4500.00 | Bonus: 450.00 | New Salary:

4950.00

Bonus updated for Department: Sales

Running UpdateEmployeeBonus...

Employee: John Doe | Old Salary: 4400.00 | Bonus: 440.00 | New Salary:

4840.00

Employee: Jane Smith | Old Salary: 4950.00 | Bonus: 495.00 | New Salary:

5445.00

Bonus updated for Department: Sales

**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.

**CODE:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER(12,2);

BEGIN

-- Check balance of source account

SELECT balance INTO v\_balance FROM accounts WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.put\_line('Insufficient balance in Account#' || p\_from\_account);

RETURN;

END IF;

-- Deduct from source

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

-- Add to destination

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

COMMIT;

DBMS\_OUTPUT.put\_line('Transferred ' || p\_amount || ' from Account#' || p\_from\_account ||

' to Account#' || p\_to\_account);

END;

/

-- Run Scenario 3 procedure: transfer $1000 from Account#202 to Account#203

BEGIN

DBMS\_OUTPUT.put\_line('Running TransferFunds...');

TransferFunds(202, 203, 1000);

END;

/

**Output:**

Running TransferFunds...

Transferred 1000 from Account#202 to Account#203