**Exercise: 1 – CONTROL STRUCTURES**

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

o 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.

o 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.

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

**Scenario 1**: Senior Discount (Age > 60)

**CODE:**

BEGIN

FOR customer IN (SELECT customer\_id, loan\_id, interest\_rate

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

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

LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 0.01

WHERE loan\_id = customer.loan\_id;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to customer ' || customer.customer\_id);

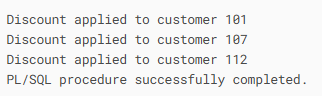
END LOOP;

COMMIT;

END;

/

**Output:**



## Scenario 2: VIP Promotion (Balance > $10,000)

**CODE:**

BEGIN

FOR customer IN (SELECT customer\_id FROM customers WHERE account\_balance > 10000)

LOOP

UPDATE customers

SET is\_vip = 'TRUE'

WHERE customer\_id = customer.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('VIP status given to customer ' || customer.customer\_id);

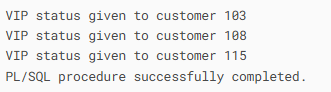
END LOOP;

COMMIT;

END;

/

**Output:**



## Scenario 3: Loan Reminders (Due in 30 Days)

**CODE:**

BEGIN

FOR loan IN (SELECT customer\_name, email, amount\_due, due\_date

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

WHERE due\_date BETWEEN SYSDATE AND SYSDATE+30)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Remind ' || loan.customer\_name ||

' at ' || loan.email ||

' to pay $' || loan.amount\_due ||

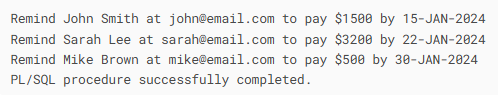
' by ' || loan.due\_date);

END LOOP;

END;

/

**Output:**



**Exercise: 3 – STORED PROCEDURES**

## Scenario 1: Process Monthly Interest (1%)

**CODE:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE savings\_accounts

SET balance = balance \* 1.01;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts');

COMMIT;

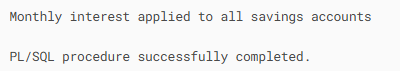
END;

/

*//Execution*

EXEC ProcessMonthlyInterest;

**Output:**



## Scenario 2: Update Employee Bonus by Department

**CODE:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

SET salary = salary \* (1 + p\_bonus\_percent/100)

WHERE department\_id = p\_dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department ' || p\_dept\_id);

COMMIT;

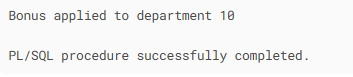
END;

/

*// Execution*

EXEC UpdateEmployeeBonus(10, 5);

**Output:**



## Scenario 3: Transfer Funds Between Accounts

**CODE:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account;

IF v\_balance >= p\_amount THEN

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: $' || p\_amount ||

' from account ' || p\_from\_account ||

' to account ' || p\_to\_account);

COMMIT;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient funds');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found');

END;

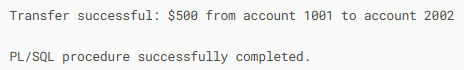
/

*// Execution-*

**CASE 1:**

EXEC TransferFunds(1001, 2002, 500);

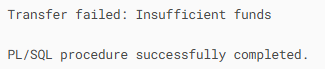
**Output:**



**CASE 2:**

EXEC TransferFunds(1001, 2002, 5000);

**Output:**



**CASE 3:**

EXEC TransferFunds(9999, 2002, 100);

**Output:**

