**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.

-- Customer Table Creation

CREATE TABLE customers (

    customer\_id NUMBER PRIMARY KEY,

    name VARCHAR2(50),

    age NUMBER(3),

    loan\_interest\_rate NUMBER(4,2)

);

INSERT INTO customers VALUES (1, 'Ram', 82, 7.8);

INSERT INTO customers VALUES (2, 'Nina', 48, 6.3);

INSERT INTO customers VALUES (3, 'Lohith', 65, 8.9);

INSERT INTO customers VALUES (4, 'Dinesh', 55, 5.1);

-- Applying Interest Rate

DECLARE

    CURSOR seniors IS

        SELECT c.customer\_id, c.loan\_interest\_rate

        FROM customers c

        WHERE c.age > 60;

BEGIN

    FOR rec IN seniors LOOP

        UPDATE customers

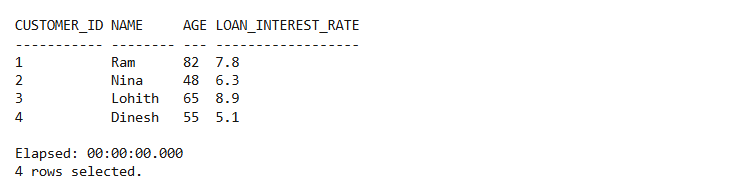
        SET loan\_interest\_rate = rec.loan\_interest\_rate - 1

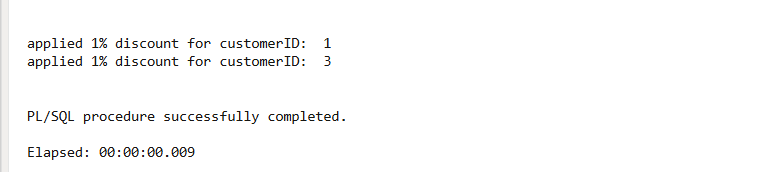
        WHERE customer\_id = rec.customer\_id;

        DBMS\_OUTPUT.PUT\_LINE('applied 1% discount for customerID:  ' || rec.customer\_id);

    END LOOP;

END;





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

* 1. **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, name, balance FROM customers ) LOOP

        IF cust.balance > 10000 THEN

            UPDATE customers

            SET is\_vip = 'Y'

            WHERE customer\_id = cust.customer\_id;

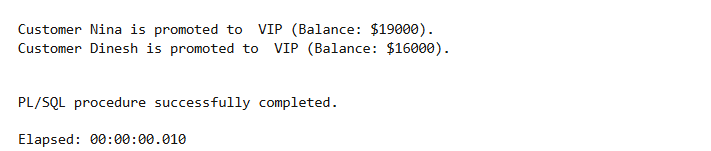
            DBMS\_OUTPUT.PUT\_LINE('Customer is ' || cust.name ||

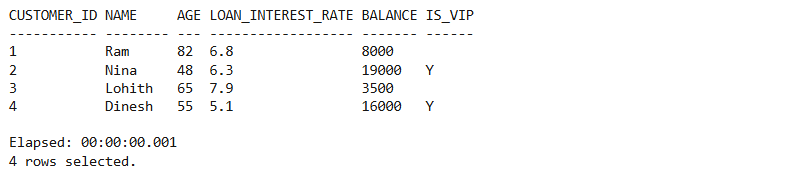
                                 ' promoted to  VIP (Balance-> $' || cust.balance || ').');

        END IF;

    END LOOP;

END;





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

* 1. **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

    FOR loan\_rec IN (

        SELECT l.loan\_id, l.due\_date, c.name

        FROM loans l

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

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

    ) LOOP

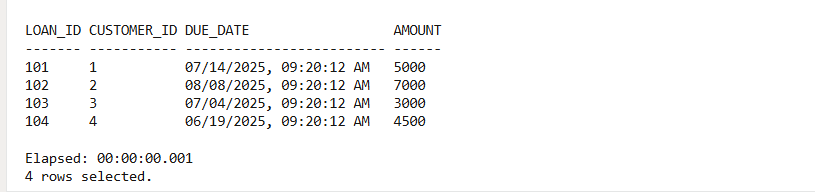
        DBMS\_OUTPUT.PUT\_LINE('Reminder--> Loan ID ' || loan\_rec.loan\_id ||

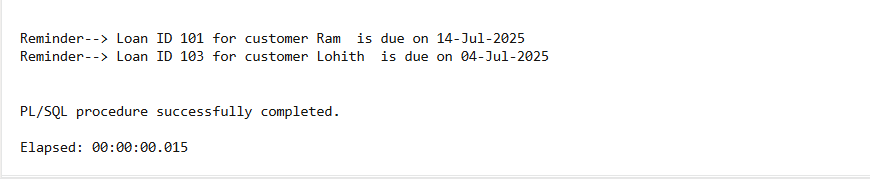
                             ' for customer ' || loan\_rec.name ||

                             '  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.

* 1. **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

    FOR rec IN (SELECT account\_id, customer\_id, balance FROM savings\_accounts) LOOP

        UPDATE savings\_accounts

        SET balance = rec.balance \* 1.01

        WHERE account\_id = rec.account\_id;

        DBMS\_OUTPUT.PUT\_LINE('1% interest is applied to Customer ID: ' || rec.customer\_id ||

                             ', Account ID-> ' || rec.account\_id);

    END LOOP;

    COMMIT;

END;

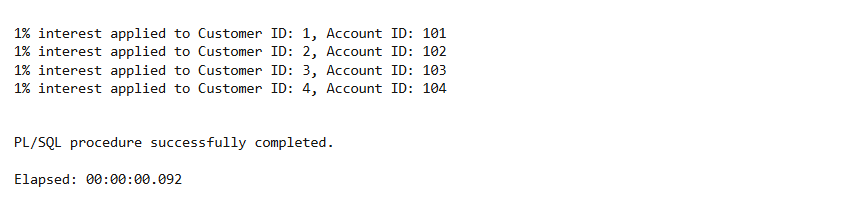
/

BEGIN

ProcessMonthlyInterest;

END;

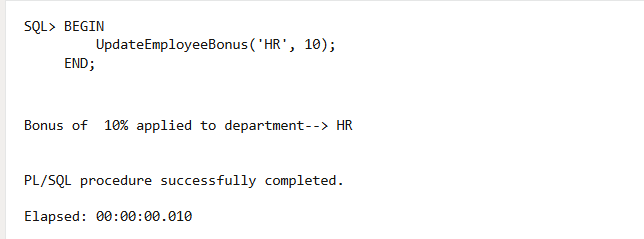


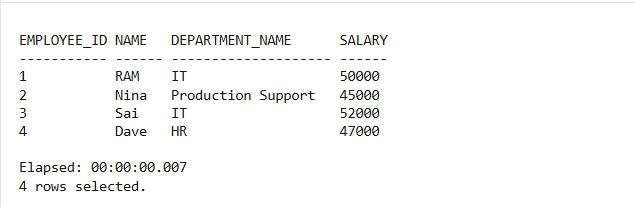


**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* 1. **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.

1. CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
2. dept\_name IN VARCHAR2,
3. bonus\_pct IN NUMBER
4. ) AS
5. BEGIN
6. UPDATE employees
7. SET salary=salary + (salary \* bonus\_pct / 100)
8. WHERE department\_name =dept\_name;
9. DBMS\_OUTPUT.PUT\_LINE('Bonus of  ' || bonus\_pct || '% applied to department--> ' || dept\_name);
10. COMMIT;
11. END;
12. /
13. BEGIN
14. UpdateEmployeeBonus('HR', 10);
15. END;
16. /





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

* 1. **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.

1. CREATE OR REPLACE PROCEDURE TransferFunds (
2. from\_account\_id IN NUMBER,
3. to\_account\_id IN NUMBER,
4. amount IN NUMBER
5. ) AS
6. v\_balance NUMBER;
7. BEGIN
8. SELECT balance INTO v\_balance
9. FROM accounts
10. WHERE account\_id = from\_account\_id;
12. IF v\_balance < amount THEN
13. DBMS\_OUTPUT.PUT\_LINE('Transfer failed--> Insufficient balance.');
14. RETURN;
15. END IF;
16. UPDATE accounts
17. SET balance = balance - amount
18. WHERE account\_id = from\_account\_id;
19. UPDATE accounts
20. SET balance = balance + amount
21. WHERE account\_id = to\_account\_id;
22. DBMS\_OUTPUT.PUT\_LINE('Transferred  ' || amount || ' from account\_ ' || from\_account\_id || ' to  ' || to\_account\_id);
23. COMMIT;
24. END;
25. /
26. EXEC TransferFunds(201, 202, 1500); --transfer 1500 from account 201 to 202

