**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.

**PROGRAM**

BEGIN

FOR rec IN (

SELECT l.loan\_id

FROM customers c

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

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

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE loan\_id = rec.loan\_id;

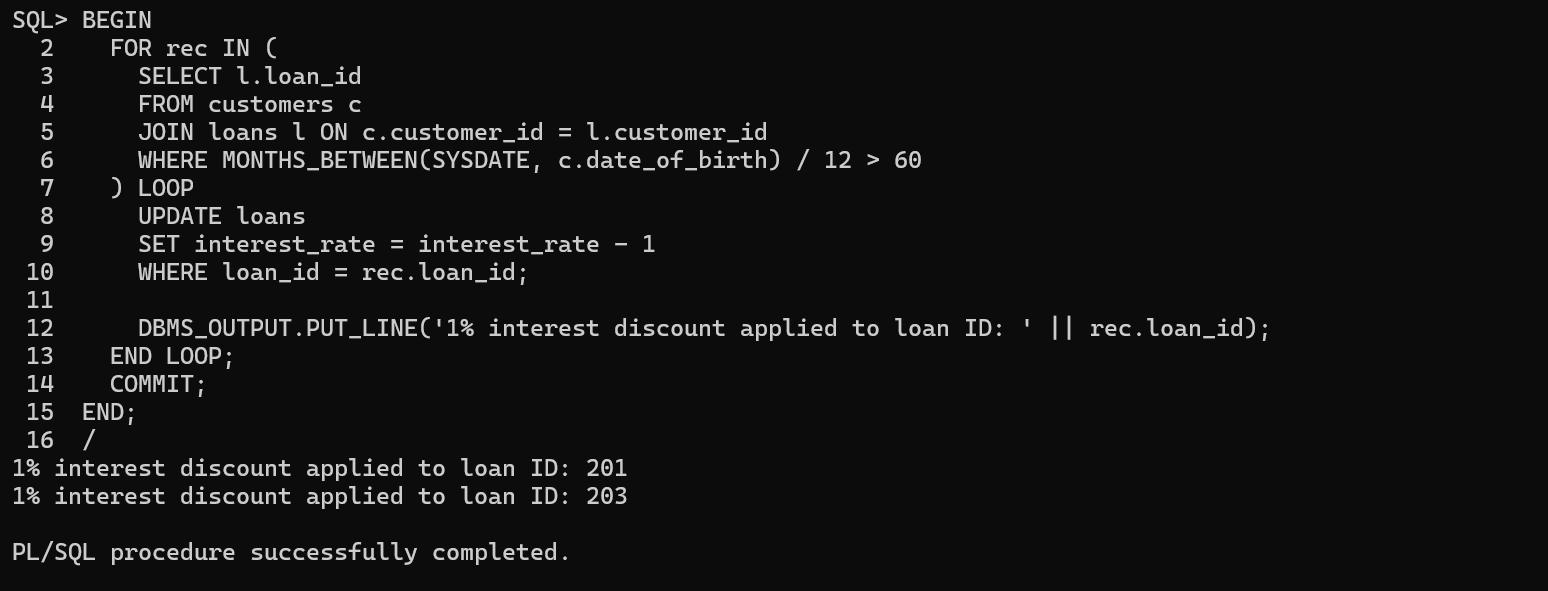
DBMS\_OUTPUT.PUT\_LINE('1% interest discount applied to loan ID: ' || rec.loan\_id);

END LOOP;

COMMIT;

END;

**OUTPUT:**



**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.

**PROGRAM:**

BEGIN

FOR rec IN (

SELECT customer\_id

FROM customers

WHERE balance > 10000

) LOOP

UPDATE customers

SET isvip = 'TRUE'

WHERE customer\_id = rec.customer\_id;

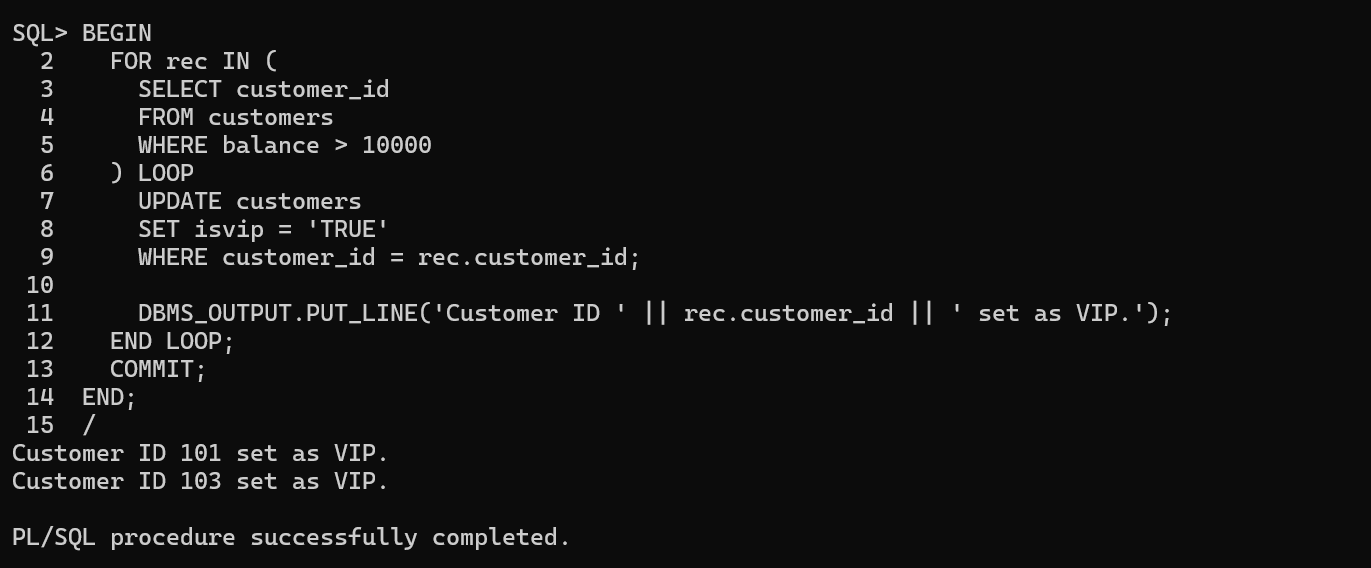
DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || rec.customer\_id || ' set as VIP.');

END LOOP;

COMMIT;

END;

**OUTPUT:**



**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.

**PROGRAM:**

BEGIN

FOR rec IN (

SELECT l.loan\_id, c.customer\_id, c.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 ID ' || rec.loan\_id ||

' for customer ' || rec.name ||

' (ID: ' || rec.customer\_id ||

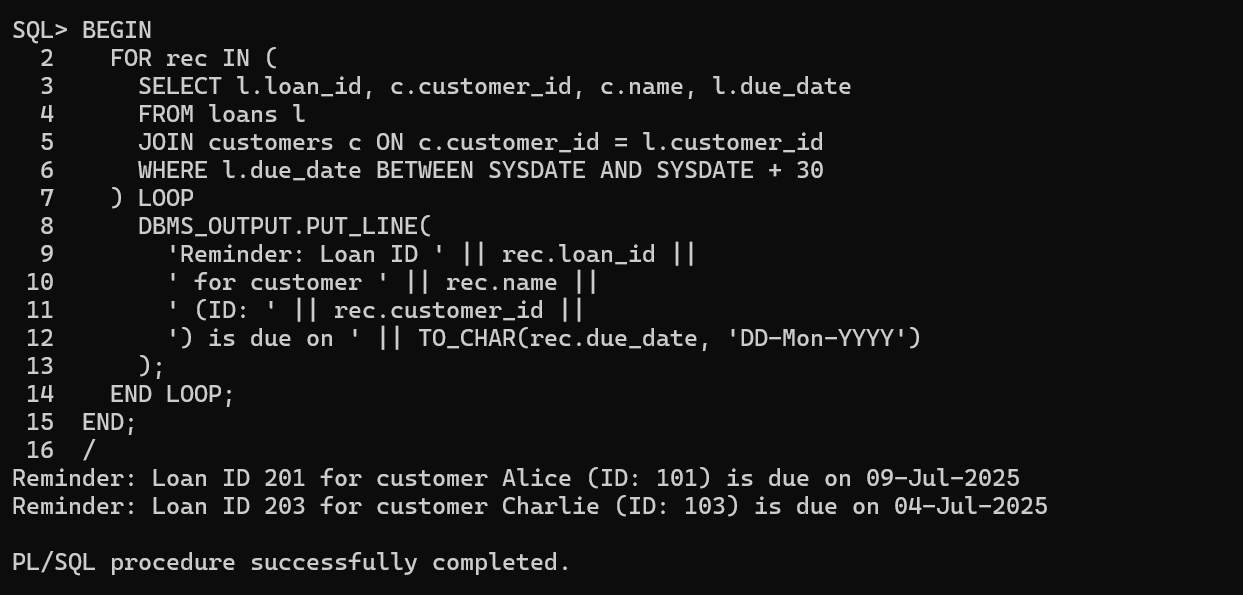
') is due on ' || TO\_CHAR(rec.due\_date, 'DD-Mon-YYYY')

);

END LOOP;

END;

**OUTPUT:**



**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.  
**PROGRAM:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

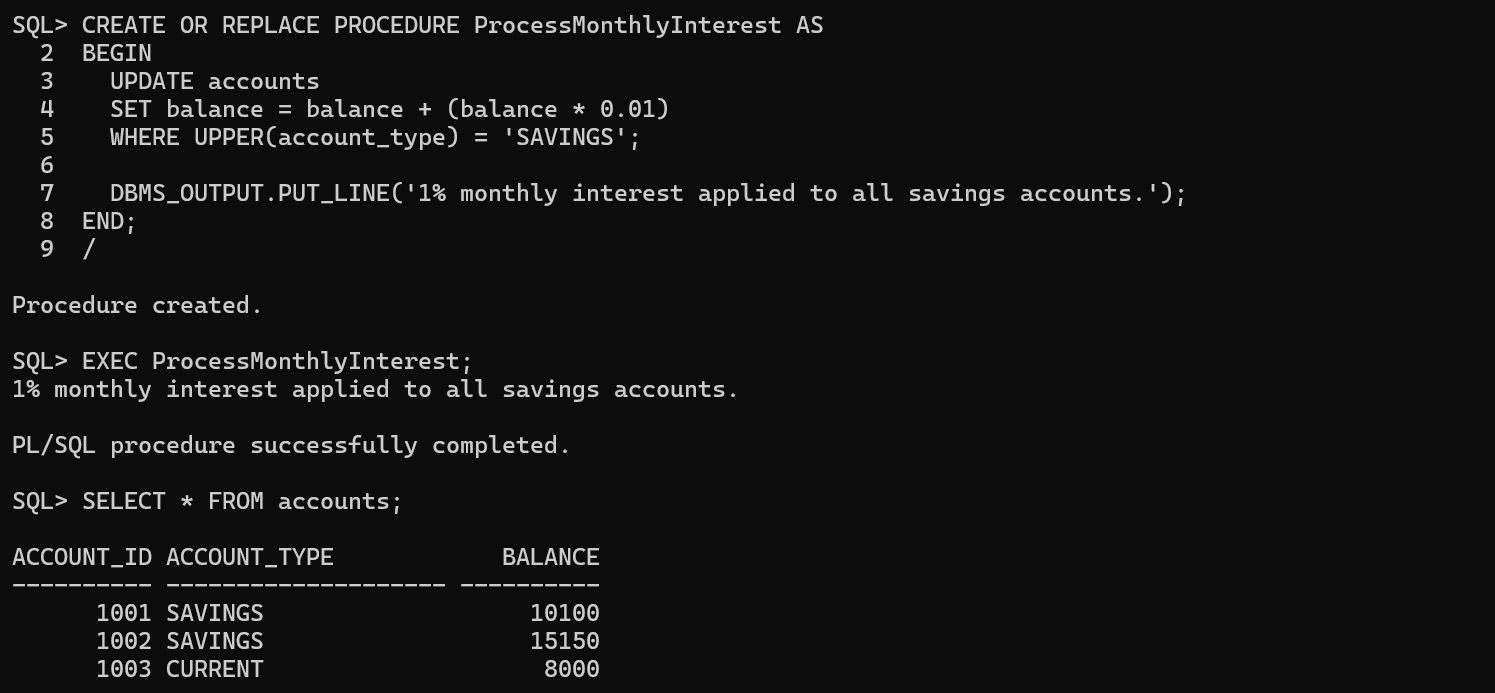
WHERE UPPER(account\_type) = 'SAVINGS';

DBMS\_OUTPUT.PUT\_LINE('1% monthly interest applied to all savings accounts.');

END;

EXEC ProcessMonthlyInterest;

**OUTPUT:**



**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.

**PROGRAM:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department\_id = dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || bonus\_percent || '% applied to department ' || dept\_id);

END;

EXEC UpdateEmployeeBonus(101, 10);

**OUTPUT:**



**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.

**PROGRAM:**

CREATE OR REPLACE PROCEDURE TransferFunds (

source\_account\_id IN NUMBER,

target\_account\_id IN NUMBER,

amount IN NUMBER

) AS

source\_balance NUMBER;

BEGIN

SELECT balance INTO source\_balance

FROM accounts

WHERE account\_id = source\_account\_id;

IF source\_balance < amount THEN

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

END IF;

UPDATE accounts SET balance = balance - amount WHERE account\_id = source\_account\_id;

UPDATE accounts SET balance = balance + amount WHERE account\_id = target\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from account ' || source\_account\_id || ' to account ' || target\_account\_id);

END;

EXEC TransferFunds(1001, 1002, 1000);

**OUTPUT:**

