**MANDATORY EXERCISES**

**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.

SET SERVEROUTPUT ON;

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

IsVIP VARCHAR2(10)

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

interest\_rate NUMBER,

due\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

INSERT INTO customers VALUES (1, 'John Doe', 65, 12000, 'FALSE');

INSERT INTO customers VALUES (2, 'Jane Smith', 45, 8000, 'FALSE');

INSERT INTO customers VALUES (3, 'David Lee', 70, 15000, 'FALSE');

INSERT INTO customers VALUES (4, 'Sara Khan', 30, 10500, 'FALSE');

INSERT INTO customers VALUES (5, 'Alex Kim', 59, 9500, 'FALSE');

INSERT INTO loans VALUES (101, 1, 9.5, SYSDATE + 10); -- due soon

INSERT INTO loans VALUES (102, 2, 8.0, SYSDATE + 40); -- not due soon

INSERT INTO loans VALUES (103, 3, 10.5, SYSDATE + 20); -- due soon

INSERT INTO loans VALUES (104, 4, 7.0, SYSDATE + 5); -- due soon

INSERT INTO loans VALUES (105, 5, 6.5, SYSDATE + 60); -- not due soon

COMMIT;

BEGIN

FOR cust IN (SELECT customer\_id, name, age FROM customers) LOOP

IF cust.age > 60 THEN

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust.customer\_id;

DBMS\_OUTPUT.PUT\_LINE(' 1% discount applied to Customer ID: ' || cust.customer\_id ||

' | Name: ' || cust.name);

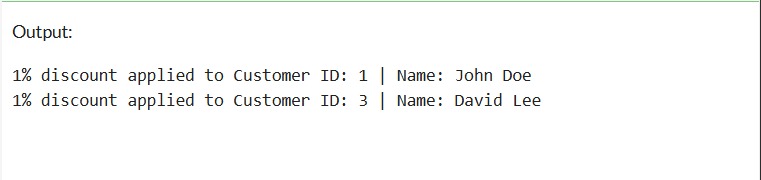
END IF;

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, name, balance FROM customers WHERE balance > 10000) LOOP

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customer\_id = cust.customer\_id;

DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust.customer\_id || ' | Name: ' || cust.name ||

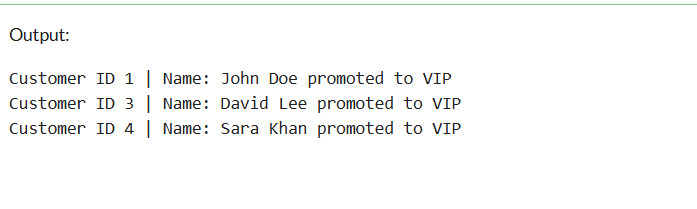
' promoted to VIP');

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.

BEGIN

FOR loan\_rec IN (

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

FROM loans l

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

WHERE l.due\_date <= SYSDATE + 30

) LOOP

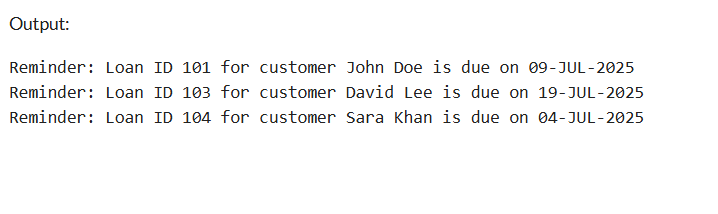
DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.loan\_id || ' for customer ' ||

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.

**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 TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

account\_type VARCHAR2(20),

balance NUMBER(10, 2)

);

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10, 2)

);

INSERT INTO accounts VALUES (101, 'Alice', 'savings', 10000);

INSERT INTO accounts VALUES (102, 'Bob', 'current', 15000);

INSERT INTO accounts VALUES (103, 'Charlie', 'savings', 8000);

INSERT INTO accounts VALUES (104, 'Diana', 'savings', 12000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'savings';

DBMS\_OUTPUT.PUT\_LINE('Interest has been added to all savings accounts (1%).');

END;

/

SET SERVEROUTPUT ON;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('ACCOUNT BALANCES BEFORE INTEREST');

FOR acc IN (SELECT \* FROM accounts ORDER BY account\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || acc.account\_id ||

', Name: ' || acc.customer\_name ||

', Type: ' || acc.account\_type ||

', Balance: ' || acc.balance

);

END LOOP;

END;

/

EXEC ProcessMonthlyInterest;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '===== ACCOUNT BALANCES AFTER INTEREST =====');

FOR acc IN (SELECT \* FROM accounts ORDER BY account\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || acc.account\_id ||

', Name: ' || acc.customer\_name ||

', Type: ' || acc.account\_type ||

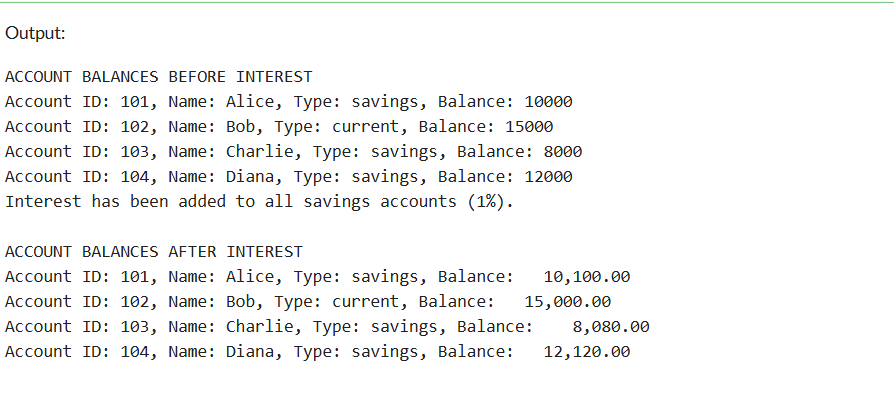
', Balance: ' || TO\_CHAR(acc.balance, '999,999.00')

);

END LOOP;

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 TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER(10, 2)

);

INSERT INTO employees VALUES (1, 'John', 'IT', 50000);

INSERT INTO employees VALUES (2, 'Jane', 'HR', 45000);

INSERT INTO employees VALUES (3, 'Sam', 'IT', 52000);

INSERT INTO employees VALUES (4, 'Ravi', 'Finance', 60000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department = p\_department;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_pct || '% applied to ' || p\_department || ' department.');

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('===== SALARIES BEFORE BONUS =====');

FOR emp IN (SELECT \* FROM employees ORDER BY emp\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Emp ID: ' || emp.emp\_id ||

', Name: ' || emp.emp\_name ||

', Department: ' || emp.department ||

', Salary: ' || emp.salary

);

END LOOP;

UpdateEmployeeBonus('IT', 10);

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '===== SALARIES AFTER BONUS =====');

FOR emp IN (SELECT \* FROM employees ORDER BY emp\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Emp ID: ' || emp.emp\_id ||

', Name: ' || emp.emp\_name ||

', Department: ' || emp.department ||

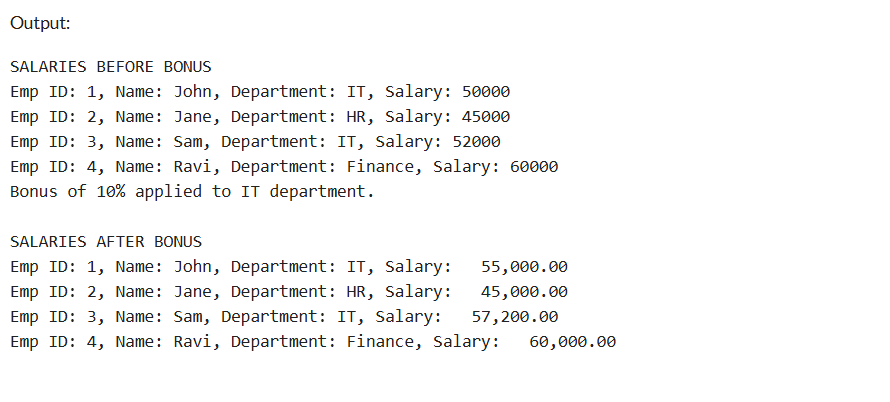
', Salary: ' || TO\_CHAR(emp.salary, '999,999.00')

);

END LOOP;

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 TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

account\_type VARCHAR2(20),

balance NUMBER(10, 2)

);

INSERT INTO accounts VALUES (101, 'Alice', 'savings', 10000);

INSERT INTO accounts VALUES (102, 'Bob', 'current', 15000);

INSERT INTO accounts VALUES (103, 'Charlie', 'savings', 8000);

INSERT INTO accounts VALUES (104, 'Diana', 'current', 12000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

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

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in Account ' || p\_from\_account);

ELSE

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(''|| p\_amount || ' transferred from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both account IDs are invalid.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('BALANCES BEFORE TRANSFER ');

FOR acc IN (SELECT \* FROM accounts ORDER BY account\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || acc.account\_id ||

', Name: ' || acc.customer\_name ||

', Type: ' || acc.account\_type ||

', Balance: ' || acc.balance

);

END LOOP;

TransferFunds(101, 102, 2000);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || 'BALANCES OF TRANSFERRED ACCOUNTS ');

FOR acc IN (SELECT \* FROM accounts WHERE account\_id IN (101, 102) ORDER BY account\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Account ID: ' || acc.account\_id ||

', Name: ' || acc.customer\_name ||

', Type: ' || acc.account\_type ||

', Balance: ' || TO\_CHAR(acc.balance, '999,999.00')

);

END LOOP;

END;

/

