PL/SQL PROGRAMMING:

EXERCISE 1: CONTROL STRUCTURES

SET SERVEROUTPUT ON;

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loans';

EXECUTE IMMEDIATE 'DROP TABLE customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

is\_vip CHAR(1) DEFAULT 'N'

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

due\_date DATE,

interest\_rate NUMBER

);

INSERT INTO customers VALUES (1, 'Arun', 65, 15000, 'N');

INSERT INTO customers VALUES (2, 'Kavita', 45, 8000, 'N');

INSERT INTO customers VALUES (3, 'Mani', 70, 12000, 'N');

INSERT INTO loans VALUES (101, 1, SYSDATE + 10, 7.5);

INSERT INTO loans VALUES (102, 2, SYSDATE + 40, 6.8);

INSERT INTO loans VALUES (103, 3, SYSDATE + 5, 8.2);

COMMIT;

Scenario 1: Apply 1% discount to interest rates for customers age > 60

BEGIN

FOR rec IN (SELECT l.loan\_id, c.age

FROM loans l

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

LOOP

IF rec.age > 60 THEN

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE loan\_id = rec.loan\_id;

END IF;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Interest rate discount applied for age > 60.');

END;

/

Scenario 2: Set IsVIP flag for balance > 10,000

BEGIN

FOR rec IN (SELECT customer\_id, balance FROM customers)

LOOP

IF rec.balance > 10000 THEN

UPDATE customers

SET is\_vip = 'Y'

WHERE customer\_id = rec.customer\_id;

END IF;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status set for customers with balance > 10000.');

END;

/

Scenario 3: Print reminders for loans due in next 30 days

BEGIN

FOR 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 <= SYSDATE + 30

)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || rec.loan\_id ||

' for customer ' || rec.name ||

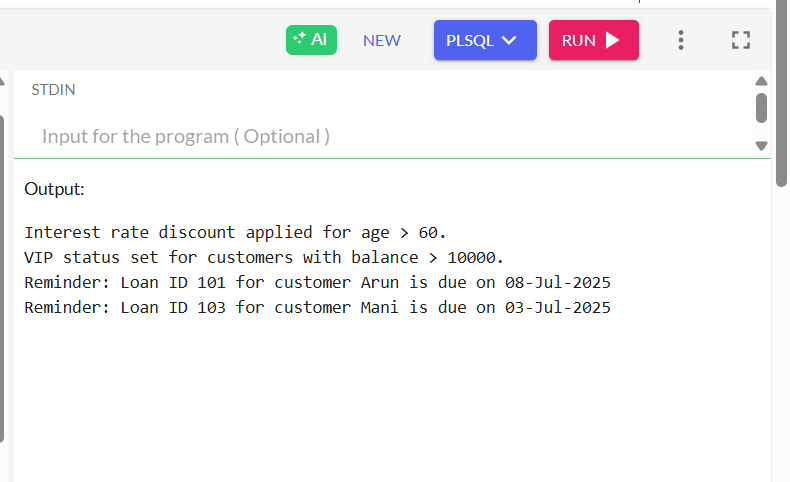
' is due on ' || TO\_CHAR(rec.due\_date, 'DD-Mon-YYYY'));

END LOOP;

END;

/

OUTPUT:



SET SERVEROUTPUT ON;

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER,

account\_type VARCHAR2(20)

);

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER

);

INSERT INTO accounts VALUES (1, 'John Doe', 10000, 'savings');

INSERT INTO accounts VALUES (2, 'Jane Smith', 5000, 'savings');

INSERT INTO accounts VALUES (3, 'Mike Brown', 3000, 'current');

INSERT INTO employees VALUES (101, 'Alice', 'HR', 50000);

INSERT INTO employees VALUES (102, 'Bob', 'IT', 60000);

INSERT INTO employees VALUES (103, 'Charlie', 'IT', 55000);

COMMIT;

Scenario 1 - Monthly Interest Procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT account\_id, balance FROM accounts WHERE account\_type = 'savings') LOOP

UPDATE accounts

SET balance = balance + (acc.balance \* 0.01)

WHERE account\_id = acc.account\_id;

END LOOP;

COMMIT;

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

END;

/

Scenario 2 - Bonus Procedure

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department = dept\_name;

COMMIT;

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

END;

/

Scenario 3 - Fund Transfer Procedure

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) AS

insufficient\_balance EXCEPTION;

BEGIN

DECLARE

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance FROM accounts WHERE account\_id = from\_account\_id;

IF from\_balance < amount THEN

RAISE insufficient\_balance;

END IF;

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount || ' completed from account ' || from\_account\_id || ' to ' || to\_account\_id);

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in account ' || from\_account\_id);

WHEN NO\_DATA\_FOUND THEN

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

END;

END;

/

BEGIN

ProcessMonthlyInterest;

UpdateEmployeeBonus('IT', 10); -- Give 10% bonus to IT department

TransferFunds(1, 2, 1000); -- Transfer 1000 from Account 1 to 2

END;

/

OUTPUT:

