**Exercise 1: Control Structures**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE loan CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customer CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

CREATE TABLE customer (

cust\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

dob DATE,

balance NUMBER(12,2),

is\_vip CHAR(1)

);

/

CREATE TABLE loan (

loan\_id NUMBER PRIMARY KEY,

cust\_id NUMBER REFERENCES customer(cust\_id),

interest\_rate NUMBER(5,2),

due\_date DATE

);

/

INSERT INTO customer VALUES (1, 'John Smith', DATE '1950-01-01', 12000, NULL);

INSERT INTO customer VALUES (2, 'Jane Doe', DATE '1985-05-10', 8000, NULL);

INSERT INTO customer VALUES (3, 'Alice King', DATE '1945-07-20', 15000, 'Y');

INSERT INTO loan VALUES (101, 1, 7.5, SYSDATE + 15);

INSERT INTO loan VALUES (102, 2, 8.0, SYSDATE + 45);

INSERT INTO loan VALUES (103, 3, 6.5, SYSDATE + 5);

COMMIT;

/

BEGIN

UPDATE loan l

SET l.interest\_rate = l.interest\_rate \* 0.99

WHERE EXISTS (

SELECT 1 FROM customer c

WHERE c.cust\_id = l.cust\_id

AND MONTHS\_BETWEEN(SYSDATE, c.dob) / 12 > 60

);

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' loan(s) discounted.');

COMMIT;

END;

/

BEGIN

UPDATE customer

SET is\_vip = 'Y'

WHERE balance > 10000 AND NVL(is\_vip, 'N') <> 'Y';

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' customer(s) promoted to VIP.');

COMMIT;

END;

/

DECLARE

CURSOR due\_loans IS

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

FROM loan l

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

WHERE l.due\_date BETWEEN TRUNC(SYSDATE) AND TRUNC(SYSDATE + 30);

BEGIN

FOR r IN due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder → ' || r.name || ' (ID ' || r.cust\_id || ') - Loan ' ||

r.loan\_id || ' due on ' || TO\_CHAR(r.due\_date, 'DD-Mon-YYYY')

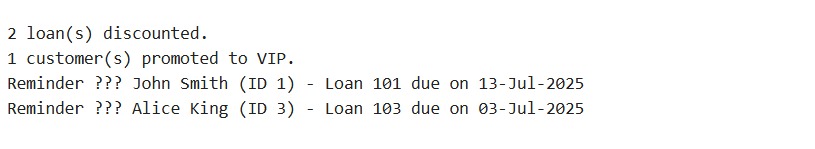
);

END LOOP;

END;

/

**Output:**



**Exercise 3: Stored Procedures**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE employees';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

account\_type VARCHAR2(20),

balance NUMBER

);

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

department\_id NUMBER,

salary NUMBER

);

INSERT INTO accounts VALUES (1, 'savings', 1000);

INSERT INTO accounts VALUES (2, 'savings', 2000);

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

INSERT INTO employees VALUES (101, 10, 3000);

INSERT INTO employees VALUES (102, 10, 3500);

INSERT INTO employees VALUES (103, 20, 4000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'savings';

END;

/

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;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_account\_id IN NUMBER,

to\_account\_id IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance

FROM accounts

WHERE account\_id = from\_account\_id;

IF from\_balance >= amount THEN

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = from\_account\_id;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = to\_account\_id;

ELSE

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

END IF;

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

BEGIN

UpdateEmployeeBonus(10, 10);

END;

/

BEGIN

TransferFunds(1, 2, 500);

END;

/

SELECT \* FROM accounts;

SELECT \* FROM employees;

**Output:**

