**EXERCISE 1:**

Program:  
SET SERVEROUTPUT ON SIZE 1000000;

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100) NOT NULL,

date\_of\_birth DATE NOT NULL,

balance NUMBER(15,2) DEFAULT 0,

is\_vip CHAR(1) DEFAULT 'N' CHECK (is\_vip IN ('Y', 'N')),

email VARCHAR2(100),

phone VARCHAR2(15),

created\_date DATE DEFAULT SYSDATE

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER NOT NULL,

loan\_amount NUMBER(15,2) NOT NULL,

interest\_rate NUMBER(5,2) NOT NULL,

loan\_start\_date DATE NOT NULL,

loan\_due\_date DATE NOT NULL,

loan\_status VARCHAR2(20) DEFAULT 'ACTIVE',

CONSTRAINT fk\_customer\_loan FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

CREATE SEQUENCE customer\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE loan\_seq START WITH 1 INCREMENT BY 1;

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'John Smith', DATE '1950-05-15', 15000.00, 'john.smith@email.com', '555-0101');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Maria Garcia', DATE '1960-08-22', 8500.00, 'maria.garcia@email.com', '555-0102');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Robert Johnson', DATE '1955-12-10', 25000.00, 'robert.johnson@email.com', '555-0103');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Emily Davis', DATE '1985-03-18', 12000.00, 'emily.davis@email.com', '555-0104');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Michael Wilson', DATE '1945-07-30', 5000.00, 'michael.wilson@email.com', '555-0105');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Sarah Brown', DATE '1992-11-05', 7500.00, 'sarah.brown@email.com', '555-0106');

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 1, 50000.00, 5.5, DATE '2024-01-15', SYSDATE + 20);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 2, 30000.00, 6.0, DATE '2024-03-10', SYSDATE + 15);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 3, 75000.00, 4.8, DATE '2024-02-20', SYSDATE + 45);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 4, 40000.00, 5.2, DATE '2024-05-05', SYSDATE + 25);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 5, 20000.00, 6.5, DATE '2024-04-12', SYSDATE + 10);

COMMIT;

DECLARE

v\_customer\_count NUMBER := 0;

v\_updated\_count NUMBER := 0;

v\_age NUMBER;

v\_loans\_updated NUMBER;

CURSOR c\_customers IS

SELECT DISTINCT c.customer\_id, c.customer\_name, c.date\_of\_birth,

FLOOR(MONTHS\_BETWEEN(SYSDATE, c.date\_of\_birth) / 12) AS age

FROM customers c

WHERE EXISTS (SELECT 1 FROM loans l WHERE l.customer\_id = c.customer\_id)

ORDER BY c.customer\_name;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('SCENARIO 1: APPLYING SENIOR CITIZEN DISCOUNT');

FOR customer\_rec IN c\_customers LOOP

v\_customer\_count := v\_customer\_count + 1;

IF customer\_rec.age > 60 THEN

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || customer\_rec.customer\_name || ' (Age: ' || customer\_rec.age || ') - ELIGIBLE for discount');

UPDATE loans

SET interest\_rate = GREATEST(interest\_rate - 1.0, 1.0)

WHERE customer\_id = customer\_rec.customer\_id

AND loan\_status = 'ACTIVE';

v\_loans\_updated := SQL%ROWCOUNT;

v\_updated\_count := v\_updated\_count + v\_loans\_updated;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || customer\_rec.customer\_name || ' (Age: ' || customer\_rec.age || ') - Not eligible');

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Total customers processed: ' || v\_customer\_count);

DBMS\_OUTPUT.PUT\_LINE('Total loans updated: ' || v\_updated\_count);

IF v\_updated\_count > 0 THEN

COMMIT;

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END;

/

DECLARE

v\_total\_customers NUMBER := 0;

v\_vip\_promoted NUMBER := 0;

v\_already\_vip NUMBER := 0;

v\_not\_eligible NUMBER := 0;

CURSOR c\_all\_customers IS

SELECT customer\_id, customer\_name, balance, is\_vip FROM customers ORDER BY balance DESC;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('SCENARIO 2: VIP STATUS PROMOTION');

FOR customer\_rec IN c\_all\_customers LOOP

v\_total\_customers := v\_total\_customers + 1;

IF customer\_rec.balance > 10000 THEN

IF customer\_rec.is\_vip = 'Y' THEN

v\_already\_vip := v\_already\_vip + 1;

ELSE

UPDATE customers SET is\_vip = 'Y' WHERE customer\_id = customer\_rec.customer\_id;

v\_vip\_promoted := v\_vip\_promoted + 1;

END IF;

ELSE

v\_not\_eligible := v\_not\_eligible + 1;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Total customers processed: ' || v\_total\_customers);

DBMS\_OUTPUT.PUT\_LINE('Newly promoted to VIP: ' || v\_vip\_promoted);

DBMS\_OUTPUT.PUT\_LINE('Already VIP customers: ' || v\_already\_vip);

DBMS\_OUTPUT.PUT\_LINE('Not eligible: ' || v\_not\_eligible);

IF v\_vip\_promoted > 0 THEN

COMMIT;

END IF;

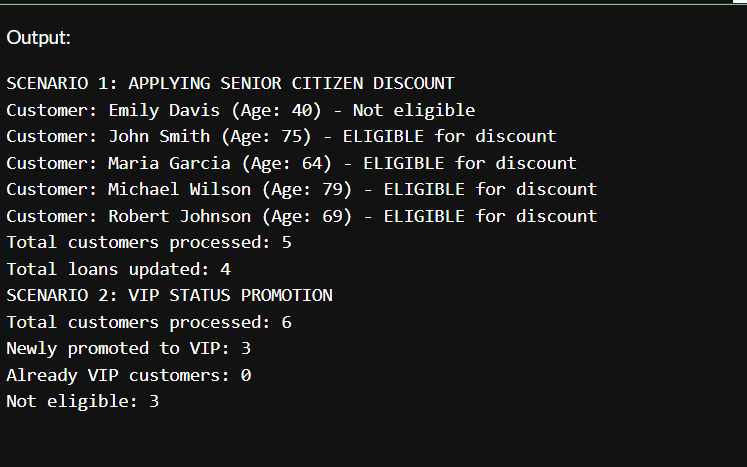
EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END;

/  
  
Output:  
  


**EXERCISE 3:**

Program:  
SET SERVEROUTPUT ON SIZE 1000000;

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100) NOT NULL,

date\_of\_birth DATE NOT NULL,

balance NUMBER(15,2) DEFAULT 0,

is\_vip CHAR(1) DEFAULT 'N' CHECK (is\_vip IN ('Y', 'N')),

email VARCHAR2(100),

phone VARCHAR2(15),

created\_date DATE DEFAULT SYSDATE

);

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER NOT NULL,

loan\_amount NUMBER(15,2) NOT NULL,

interest\_rate NUMBER(5,2) NOT NULL,

loan\_start\_date DATE NOT NULL,

loan\_due\_date DATE NOT NULL,

loan\_status VARCHAR2(20) DEFAULT 'ACTIVE',

CONSTRAINT fk\_customer\_loan FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

CREATE SEQUENCE customer\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE loan\_seq START WITH 1 INCREMENT BY 1;

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'John Smith', DATE '1950-05-15', 15000.00, 'john.smith@email.com', '555-0101');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Maria Garcia', DATE '1960-08-22', 8500.00, 'maria.garcia@email.com', '555-0102');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Robert Johnson', DATE '1955-12-10', 25000.00, 'robert.johnson@email.com', '555-0103');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Emily Davis', DATE '1985-03-18', 12000.00, 'emily.davis@email.com', '555-0104');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Michael Wilson', DATE '1945-07-30', 5000.00, 'michael.wilson@email.com', '555-0105');

INSERT INTO customers (customer\_id, customer\_name, date\_of\_birth, balance, email, phone) VALUES

(customer\_seq.NEXTVAL, 'Sarah Brown', DATE '1992-11-05', 7500.00, 'sarah.brown@email.com', '555-0106');

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 1, 50000.00, 5.5, DATE '2024-01-15', SYSDATE + 20);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 2, 30000.00, 6.0, DATE '2024-03-10', SYSDATE + 15);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 3, 75000.00, 4.8, DATE '2024-02-20', SYSDATE + 45);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 4, 40000.00, 5.2, DATE '2024-05-05', SYSDATE + 25);

INSERT INTO loans (loan\_id, customer\_id, loan\_amount, interest\_rate, loan\_start\_date, loan\_due\_date) VALUES

(loan\_seq.NEXTVAL, 5, 20000.00, 6.5, DATE '2024-04-12', SYSDATE + 10);

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

employee\_name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10,2)

);

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER(15,2)

);

INSERT INTO employees VALUES (1, 'Alice', 1, 50000);

INSERT INTO employees VALUES (2, 'Bob', 1, 45000);

INSERT INTO employees VALUES (3, 'Charlie', 2, 60000);

INSERT INTO accounts VALUES (1001, 1, 2000);

INSERT INTO accounts VALUES (1002, 2, 1000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE customers

SET balance = balance + (balance \* 0.01)

WHERE balance > 0;

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

COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department\_id = p\_dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to department ' || p\_dept\_id);

COMMIT;

END;

/

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

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

END IF;

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('Transferred ' || p\_amount || ' from account ' || p\_from\_account || ' to account ' || p\_to\_account);

COMMIT;

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

SELECT customer\_id, customer\_name, balance FROM customers;

BEGIN

UpdateEmployeeBonus(1, 10);

END;

/

SELECT employee\_id, employee\_name, department\_id, salary FROM employees;

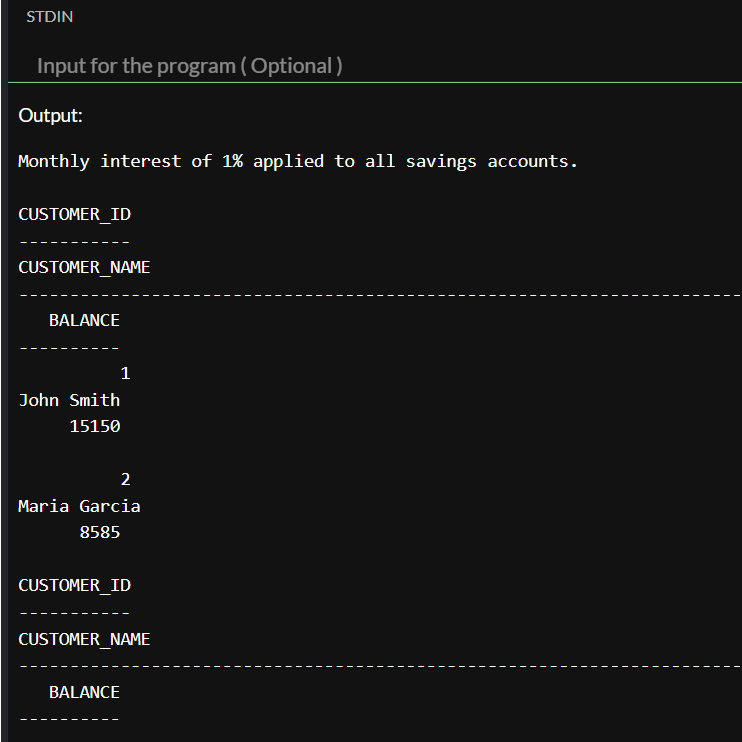
BEGIN

TransferFunds(1001, 1002, 500);

END;

/

SELECT account\_id, customer\_id, balance FROM accounts;

  
output: