DO $$

BEGIN

IF EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'transactions') THEN

EXECUTE 'DROP TABLE transactions CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'loans') THEN

EXECUTE 'DROP TABLE loans CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'accounts') THEN

EXECUTE 'DROP TABLE accounts CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'customers') THEN

EXECUTE 'DROP TABLE customers CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM information\_schema.tables WHERE table\_name = 'employees') THEN

EXECUTE 'DROP TABLE employees CASCADE';

END IF;

END

$$;

CREATE TABLE Customers (

CustomerID INTEGER PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance NUMERIC,

LastModified DATE,

IsVIP BOOLEAN DEFAULT FALSE

);

CREATE TABLE Accounts (

AccountID INTEGER PRIMARY KEY,

CustomerID INTEGER REFERENCES Customers(CustomerID),

AccountType VARCHAR(20),

Balance NUMERIC,

LastModified DATE

);

CREATE TABLE Transactions (

TransactionID INTEGER PRIMARY KEY,

AccountID INTEGER REFERENCES Accounts(AccountID),

TransactionDate DATE,

Amount NUMERIC,

TransactionType VARCHAR(10)

);

CREATE TABLE Loans (

LoanID INTEGER PRIMARY KEY,

CustomerID INTEGER REFERENCES Customers(CustomerID),

LoanAmount NUMERIC,

InterestRate NUMERIC,

StartDate DATE,

EndDate DATE

);

CREATE TABLE Employees (

EmployeeID INTEGER PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary NUMERIC,

Department VARCHAR(50),

HireDate DATE

);

INSERT INTO Customers VALUES

(1, 'Arjun', DATE '1993-04-10', 3500, CURRENT\_DATE, FALSE),

(2, 'Meera', DATE '1987-09-15', 22000, CURRENT\_DATE, FALSE),

(3, 'Kabir', DATE '1955-02-02', 700, CURRENT\_DATE, TRUE);

INSERT INTO Accounts VALUES

(1, 1, 'Savings', 3500, CURRENT\_DATE),

(2, 2, 'Checking', 22000, CURRENT\_DATE),

(3, 3, 'Savings', 700, CURRENT\_DATE);

INSERT INTO Transactions VALUES

(1, 1, CURRENT\_DATE, 250, 'Deposit'),

(2, 2, CURRENT\_DATE, 500, 'Withdrawal');

INSERT INTO Loans VALUES

(1, 1, 8000, 4.5, CURRENT\_DATE, CURRENT\_DATE + INTERVAL '48 months'),

(2, 3, 6000, 5.5, CURRENT\_DATE, CURRENT\_DATE + INTERVAL '30 days');

INSERT INTO Employees VALUES

(1, 'Riya', 'Lead', 75000, 'Admin', DATE '2016-07-10'),

(2, 'Karthik', 'Engineer', 62000, 'Development', DATE '2018-02-14');

CREATE OR REPLACE FUNCTION ProcessMonthlyInterest()

RETURNS VOID AS $$

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.015),

LastModified = CURRENT\_DATE

WHERE AccountType = 'Savings';

RAISE NOTICE 'Monthly interest (1.5%%) applied to all savings accounts.';

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION UpdateEmployeeBonus(p\_department TEXT, p\_bonus\_percent NUMERIC)

RETURNS VOID AS $$

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

RAISE NOTICE 'Bonus of % applied to department: %', p\_bonus\_percent, p\_department;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION TransferFunds(p\_from\_account INT, p\_to\_account INT, p\_amount NUMERIC)

RETURNS VOID AS $$

DECLARE

v\_from\_balance NUMERIC;

BEGIN

SELECT Balance INTO v\_from\_balance FROM Accounts WHERE AccountID = p\_from\_account;

IF v\_from\_balance < p\_amount THEN

RAISE EXCEPTION 'Insufficient funds in account %', p\_from\_account;

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = CURRENT\_DATE

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = CURRENT\_DATE

WHERE AccountID = p\_to\_account;

RAISE NOTICE 'Transferred % from account % to account %', p\_amount, p\_from\_account, p\_to\_account;

EXCEPTION

WHEN OTHERS THEN

RAISE NOTICE 'Error: %', SQLERRM;

RAISE;

END;

$$ LANGUAGE plpgsql;

DO $$

BEGIN

PERFORM ProcessMonthlyInterest();

PERFORM UpdateEmployeeBonus('Development', 7);

PERFORM TransferFunds(1, 2, 300);

END;

$$;

DO $$

DECLARE

rec RECORD;

BEGIN

RAISE NOTICE E'\n--- Accounts Balances After Updates ---';

RAISE NOTICE '%', 'AccountID | CustomerID | Type | Balance | LastModified';

FOR rec IN SELECT \* FROM Accounts ORDER BY AccountID LOOP

RAISE NOTICE '%',

LPAD(rec.AccountID::TEXT, 9) || ' | ' ||

LPAD(rec.CustomerID::TEXT, 11) || ' | ' ||

RPAD(rec.AccountType, 8) || ' | ' ||

TO\_CHAR(rec.Balance, '9999990.00') || ' | ' ||

TO\_CHAR(rec.LastModified, 'DD-Mon-YYYY');

END LOOP;

END;

$$;

DO $$

DECLARE

rec RECORD;

BEGIN

RAISE NOTICE E'\n--- Employees Salaries After Bonus ---';

RAISE NOTICE '%', 'EmployeeID | Name | Department | Salary';

FOR rec IN SELECT \* FROM Employees ORDER BY EmployeeID LOOP

RAISE NOTICE '%',

LPAD(rec.EmployeeID::TEXT, 10) || ' | ' ||

RPAD(rec.Name, 17) || ' | ' ||

RPAD(rec.Department, 12) || ' | ' ||

TO\_CHAR(rec.Salary, '9999990.00');

END LOOP;

END;

$$;