DO $$

BEGIN

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'transactions') THEN

EXECUTE 'DROP TABLE transactions CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'accounts') THEN

EXECUTE 'DROP TABLE accounts CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'loans') THEN

EXECUTE 'DROP TABLE loans CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'employees') THEN

EXECUTE 'DROP TABLE employees CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'customers') THEN

EXECUTE 'DROP TABLE customers CASCADE';

END IF;

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'transactions\_seq') THEN

EXECUTE 'DROP SEQUENCE transactions\_seq';

END IF;

IF EXISTS (SELECT 1 FROM pg\_class WHERE relname = 'log\_table') THEN

EXECUTE 'DROP TABLE log\_table';

END IF;

END $$;

CREATE TABLE Customers (

CustomerID INTEGER PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance NUMERIC,

LastModified DATE

);

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(20)

);

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

);

CREATE SEQUENCE Transactions\_seq START WITH 5;

CREATE TABLE Log\_Table (

LogTime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

Message TEXT

);

INSERT INTO Customers VALUES (1, 'Aarav', '1992-03-10', 5000, CURRENT\_DATE);

INSERT INTO Customers VALUES (2, 'Emily', '1988-12-05', 12000, CURRENT\_DATE);

INSERT INTO Customers VALUES (3, 'Raj', '1975-07-25', 800, CURRENT\_DATE);

INSERT INTO Accounts VALUES (1, 1, 'Checking', 5000, CURRENT\_DATE);

INSERT INTO Accounts VALUES (2, 2, 'Savings', 12000, CURRENT\_DATE);

INSERT INTO Accounts VALUES (3, 3, 'Savings', 800, CURRENT\_DATE);

INSERT INTO Transactions VALUES (1, 1, CURRENT\_DATE, 300, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, CURRENT\_DATE, 400, 'Deposit');

INSERT INTO Transactions VALUES (3, 1, CURRENT\_DATE, 150, 'Withdrawal');

INSERT INTO Transactions VALUES (4, 3, CURRENT\_DATE, 100, 'Deposit');

INSERT INTO Loans VALUES (1, 2, 7000, 4.5, CURRENT\_DATE, CURRENT\_DATE + INTERVAL '40 days');

INSERT INTO Loans VALUES (2, 1, 3000, 5.0, CURRENT\_DATE, CURRENT\_DATE + INTERVAL '60 days');

INSERT INTO Employees VALUES (1, 'Maya', 'Analyst', 50000, 'Finance', '2018-01-20');

INSERT INTO Employees VALUES (2, 'Liam', 'Designer', 55000, 'Marketing', '2019-09-10');

INSERT INTO Employees VALUES (3, 'Noah', 'Lead Dev', 80000, 'IT', '2016-04-15');

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

IN p\_from\_account INTEGER,

IN p\_to\_account INTEGER,

IN p\_amount NUMERIC

)

LANGUAGE plpgsql AS $$

DECLARE

v\_balance NUMERIC;

BEGIN

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

IF v\_balance < p\_amount THEN

INSERT INTO Log\_Table(Message) VALUES ('Transfer failed: Insufficient funds');

RETURN;

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;

INSERT INTO Transactions VALUES (nextval('Transactions\_seq'), p\_from\_account, CURRENT\_DATE, p\_amount, 'Transfer-Out');

INSERT INTO Transactions VALUES (nextval('Transactions\_seq'), p\_to\_account, CURRENT\_DATE, p\_amount, 'Transfer-In');

INSERT INTO Log\_Table(Message) VALUES ('Transfer successful.');

EXCEPTION

WHEN OTHERS THEN

INSERT INTO Log\_Table(Message) VALUES ('Transfer failed: ' || SQLERRM);

END

$$;

CREATE OR REPLACE PROCEDURE UpdateSalary(

IN p\_emp\_id INTEGER,

IN p\_percent NUMERIC

)

LANGUAGE plpgsql AS $$

BEGIN

UPDATE Employees

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

WHERE EmployeeID = p\_emp\_id;

IF NOT FOUND THEN

INSERT INTO Log\_Table(Message) VALUES ('Salary update failed: Employee not found');

RETURN;

END IF;

INSERT INTO Log\_Table(Message) VALUES ('Salary updated.');

EXCEPTION

WHEN OTHERS THEN

INSERT INTO Log\_Table(Message) VALUES ('Salary update failed: ' || SQLERRM);

END

$$;

CREATE OR REPLACE PROCEDURE AddNewCustomer(

IN p\_customer\_id INTEGER,

IN p\_name VARCHAR,

IN p\_dob DATE,

IN p\_balance NUMERIC

)

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO Customers VALUES (p\_customer\_id, p\_name, p\_dob, p\_balance, CURRENT\_DATE);

INSERT INTO Log\_Table(Message) VALUES ('Customer added.');

EXCEPTION

WHEN unique\_violation THEN

INSERT INTO Log\_Table(Message) VALUES ('Customer ID already exists.');

WHEN OTHERS THEN

INSERT INTO Log\_Table(Message) VALUES ('Failed to add customer: ' || SQLERRM);

END

$$;

CALL SafeTransferFunds(2, 1, 2000);

CALL SafeTransferFunds(3, 1, 5000);

CALL UpdateSalary(2, 15);

CALL UpdateSalary(99, 10);

CALL AddNewCustomer(4, 'Sophia', '1998-11-22', 3000);

CALL AddNewCustomer(1, 'Ava', '2001-01-01', 5000);

SELECT \* FROM Log\_Table ORDER BY LogTime;

SELECT \* FROM Customers;

SELECT \* FROM Accounts;

SELECT \* FROM Transactions;

SELECT \* FROM Employees;