**Exercise 6: Configuring Beans with Annotations**

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(100)

);

CREATE TABLE transactions (

transaction\_id INT PRIMARY KEY,

customer\_id INT,

transaction\_date DATE,

amount DECIMAL(10, 2),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

INSERT INTO customers (customer\_id, customer\_name) VALUES

(1, 'John Doe'),

(2, 'Jane Smith');

INSERT INTO transactions (transaction\_id, customer\_id, transaction\_date, amount) VALUES

(1, 1, '2024-08-01', 150.00),

(2, 2, '2024-08-02', 200.00);

DELIMITER //

**Scenario 1 :**

CREATE PROCEDURE GenerateMonthlyStatements()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE cust\_name VARCHAR(100);

DECLARE tran\_id INT;

DECLARE tran\_date DATE;

DECLARE tran\_amount DECIMAL(10, 2);

DECLARE cur CURSOR FOR

SELECT c.customer\_id, c.customer\_name, t.transaction\_id, t.transaction\_date, t.amount

FROM customers c

JOIN transactions t ON c.customer\_id = t.customer\_id

WHERE MONTH(t.transaction\_date) = MONTH(CURDATE()) AND YEAR(t.transaction\_date) = YEAR(CURDATE());

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO cust\_id, cust\_name, tran\_id, tran\_date, tran\_amount;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT CONCAT('Customer: ', cust\_name, ', Transaction ID: ', tran\_id, ', Date: ', tran\_date, ', Amount: ', tran\_amount) AS Statement;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

CALL GenerateMonthlyStatements();

CREATE TABLE accounts (

account\_id INT PRIMARY KEY,

customer\_id INT,

balance DECIMAL(10, 2),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

CREATE TABLE loans (

loan\_id INT PRIMARY KEY,

customer\_id INT,

loan\_amount DECIMAL(10, 2),

interest\_rate DECIMAL(5, 2),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

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

(1, 1, 5000.00, 5.00),

(2, 2, 10000.00, 4.50);

DELIMITER //

**Scenario 2:**

CREATE PROCEDURE ApplyAnnualFee()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE acc\_id INT;

DECLARE acc\_balance DECIMAL(10, 2);

DECLARE annual\_fee DECIMAL(10, 2) DEFAULT 50.00;

DECLARE cur CURSOR FOR

SELECT account\_id, balance FROM accounts;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO acc\_id, acc\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

SET acc\_balance = acc\_balance - annual\_fee;

UPDATE accounts SET balance = acc\_balance WHERE account\_id = acc\_id;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

CALL ApplyAnnualFee();

DELIMITER //

**Scenario 3:**

CREATE PROCEDURE UpdateLoanInterestRates()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE loan\_id INT;

DECLARE current\_rate DECIMAL(5, 2);

DECLARE new\_rate DECIMAL(5, 2);

DECLARE cur CURSOR FOR

SELECT loan\_id, interest\_rate FROM loans;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO loan\_id, current\_rate;

IF done THEN

LEAVE read\_loop;

END IF;

-- Assuming new policy: Increase interest rate by 1%

SET new\_rate = current\_rate + 1.00;

UPDATE loans SET interest\_rate = new\_rate WHERE loan\_id = loan\_id;

END LOOP;

CLOSE cur;

END //

DELIMITER ;

CALL UpdateLoanInterestRates();