```sql
-- ==============================================
-- Create Table
-- ==============================================
CREATE TABLE StudentEnrollments (
   student_id INT PRIMARY KEY,
   student_name VARCHAR(100) NOT NULL,
   course_id VARCHAR(10) NOT NULL,
   enrollment_date DATE NOT NULL
);
-- Insert initial data
INSERT INTO StudentEnrollments (student_id, student_name, course_id, enrollment_date)
VALUES
(1, 'Ashish', 'CSE101', '2024-06-01'),
(2, 'Smaran', 'CSE102', '2024-06-01'),
(3, 'Vaibhav', 'CSE103', '2024-06-01');
SELECT * FROM StudentEnrollments;

-- ==============================================
-- Part A: Simulate Deadlock Between Two Users
-- ==============================================
-- (Run in two sessions to cause a deadlock)
-- User A
```

```
START TRANSACTION;
UPDATE StudentEnrollments
SET enrollment_date = '2024-07-01'
WHERE student_id = 1;
 -- Locks row 1
-- Later in same transaction
UPDATE StudentEnrollments
SET enrollment_date = '2024-07-02'
WHERE student_id = 2;
-- Will wait for User B
-- User B
START TRANSACTION;
UPDATE StudentEnrollments
SET enrollment_date = '2024-07-05'
WHERE student_id = 2;
-- Locks row 2
-- Later in same transaction
UPDATE StudentEnrollments
SET enrollment_date = '2024-07-06'
WHERE student_id = 1;
-- Deadlock occurs here
-- Result: DB detects deadlock and rolls back one transaction
automatically.
```

```sql
-- ============================================

-- Part B: MVCC – Concurrent Reads and Writes

-- ============================================

-- User A (Reader)

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

SELECT * FROM StudentEnrollments WHERE student_id = 1;

-- Sees old value: enrollment_date = '2024-06-01'

-- User B (Writer)

START TRANSACTION;

UPDATE StudentEnrollments

SET enrollment_date = '2024-07-10'

WHERE student_id = 1;

COMMIT;


-- User A (still in same transaction, sees old snapshot)

SELECT * FROM StudentEnrollments WHERE student_id = 1;

-- Still sees '2024-06-01'

COMMIT;


-- New session/User A after commit

SELECT * FROM StudentEnrollments WHERE student_id = 1;

-- Now sees updated '2024-07-10'
-- ==============================================
-- Part C: Compare Locking vs MVCC
-- ==============================================
-- Without MVCC (Locking with SELECT FOR UPDATE)
-- User A
START TRANSACTION;
SELECT * FROM StudentEnrollments
WHERE student_id = 1
FOR UPDATE;
-- Locks row
-- User B
START TRANSACTION;
SELECT * FROM StudentEnrollments WHERE student_id = 1;
-- This blocks until User A commits
-- With MVCC (Non-blocking Reads)
-- User A
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION;
SELECT * FROM StudentEnrollments WHERE student_id = 1;
-- Sees old value (snapshot)
-- User B
START TRANSACTION;
UPDATE StudentEnrollments
```

```
SET enrollment_date = '2024-07-15'

WHERE student_id = 1;

COMMIT;


-- User A (still in same transaction)

SELECT * FROM StudentEnrollments WHERE student_id = 1;

-- Still sees old snapshot value

COMMIT;

-- User A (new transaction after commit)

SELECT * FROM StudentEnrollments WHERE student_id = 1;

-- Now sees updated '2024-07-15'
```