**LAB 3**

Author:

|  |  |
| --- | --- |
| **Abhirup Ranjan** | **110091866** |

Submitted To:

|  |
| --- |
| **Professor** |
| **Dr. Shafaq Khan** |

**Table of Contents**

[1. Part 1: 2](#_Toc139315896)

# **Part 1:**

1. **Create database <First\_Name>ADTLab3.**

CREATE DATABASE AbhirupADTLab3;

A screenshot of a computer

Description automatically generated with medium confidence

1. **Create table <First\_Name>Student table, <First\_Name>StudentRegistration table and <First\_Name>Course table with the following schema. Insert a few rows in the Students and Course table only.**

CREATE TABLE AbhirupStudent (

StudentID INT IDENTITY(1,1) PRIMARY KEY,

FullName VARCHAR(100),

Email VARCHAR(100),

TotalCredits INT

);

CREATE TABLE AbhirupCourses (

CourseID INT IDENTITY(1,1) PRIMARY KEY,

CourseName VARCHAR(100),

Instructor VARCHAR(100),

CourseCredits INT,

AvailableSeats INT

);

CREATE TABLE AbhirupStudentRegistration (

RegistrationID INT IDENTITY(1,1) PRIMARY KEY,

StudentID INT,

CourseID INT,

FOREIGN KEY (StudentID) REFERENCES AbhirupStudent(StudentID),

FOREIGN KEY (CourseID) REFERENCES AbhirupCourses(CourseID),

);

**A screenshot of a computer

Description automatically generated with medium confidence**

-- Insert value into Student table

INSERT INTO AbhirupStudent (FullName, Email, TotalCredits)

VALUES

('John Doe', 'john.doe@example.com', 0),

( 'Jane Smith', 'jane.smith@example.com', 0),

('Michael Johnson', 'michael.johnson@example.com', 0)

-- Insert value into Courses table

INSERT INTO AbhirupCourses (CourseName, Instructor, CourseCredits, AvailableSeats)

VALUES

('Mathematics', 'Professor Anderson', 3, 1),

('English Literature', 'Professor Thompson', 4, 20),

('Computer Science', 'Professor Roberts', 5, 10)

-- Insert value into Courses Registration table

INSERT INTO AbhirupStudentRegistration (StudentID, CourseID)

VALUES

(1,1),

(2,2),

(3,3);

**A screenshot of a computer

Description automatically generated with low confidence**

1. **We need to create a store procedure whenever a student registers for a course. The procedure must check the availability of seats in the course before registering the student for the course. If the student is registered the availability of the seats should be deducted and the credits of the course should be added to the student's total credits. The structure of the stored procedure is as follows:**

**Name of the procedure: <Your\_First\_Name>\_spInsertStudentRegistration which takes StudentID and CourseID as input parameters. - Check the availability of Seats in the provided course table. - Decrease the Availability of the Seats in the Courses Table. - Add Course credits of the Courses table to the Student Total credits in the Students table. - Insert the record into the StudentRegistration table with RegistrationID, StudentID, and CourseID. - If the available seats are less than or equal to 0 then the transaction should be rolled back and print the message ‘Course is full. Registration failed’.**

-- Create the stored procedure

CREATE PROCEDURE abhirup\_spInsertStudentRegistration

@StudentID INT,

@CourseID INT

AS

BEGIN

-- Start a transaction

BEGIN TRANSACTION;

-- Check the availability of seats in the course

DECLARE @AvailableSeats INT;

SELECT @AvailableSeats = AvailableSeats

FROM AbhirupCourses

WHERE CourseID = @CourseID;

-- If available seats are greater than 0, register the student

IF @AvailableSeats > 0

BEGIN

-- Decrease the availability of seats in the Courses table

UPDATE AbhirupCourses

SET AvailableSeats = AvailableSeats - 1

WHERE CourseID = @CourseID;

-- Add course credits to the student's total credits

DECLARE @CourseCredits INT;

SELECT @CourseCredits = CourseCredits

FROM AbhirupCourses

WHERE CourseID = @CourseID;

UPDATE AbhirupStudent

SET TotalCredits = TotalCredits + @CourseCredits

WHERE StudentID = @StudentID;

-- Insert the record into the StudentRegistration table

INSERT INTO AbhirupStudentRegistration (StudentID, CourseID)

VALUES (@StudentID, @CourseID);

-- Commit the transaction

COMMIT;

PRINT 'Registration successful.';

END

ELSE

BEGIN

-- Rollback the transaction and print an error message

ROLLBACK;

PRINT 'Course is full. Registration failed.';

END

END;

**A screenshot of a computer

Description automatically generated with medium confidence**

1. **Testing the solution by registering below students for the following course.**

**a. Jane Smith registers for Mathematics.**

**b. Michael Johnson registers for Computer Science.**

**c. John Doe registers for Mathematics.**

**All 3 Tables Before the execution:**

A screenshot of a computer

Description automatically generated with medium confidence

All 3 tables After the Execution

-- Registering Jane Smith for Mathematics

EXEC abhirup\_spInsertStudentRegistration @StudentID = 2, @CourseID = 1;

A screenshot of a computer

Description automatically generated with medium confidence

-- Registering Michael Johnson for Computer Science

EXEC abhirup\_spInsertStudentRegistration @StudentID = 3, @CourseID = 3;

A screenshot of a computer

Description automatically generated with medium confidence

-- Registering John Doe for Mathematics

EXEC abhirup\_spInsertStudentRegistration @StudentID = 1, @CourseID = 1;

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence