```
CREATE DATABASE Library;
USE Library;

-- Table: Books
CREATE TABLE Books (
    BookID CHAR(5) PRIMARY KEY CHECK (BookID LIKE 'B____'),
    Title VARCHAR(100) NOT NULL,
    Author VARCHAR(100) NOT NULL,
    Genre VARCHAR(50) CHECK (Genre IN ('Fiction', 'Non-fiction', 'Science', 'History')),
    Price DECIMAL(10, 2) CHECK (Price >= 0),
    CopiesInStock INT CHECK (CopiesInStock >= 0)
);

-- Table: Members
CREATE TABLE Members (
    MemberID CHAR(5) PRIMARY KEY CHECK (MemberID LIKE 'M____'),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    City VARCHAR(50) DEFAULT 'Unknown',
    TotalBooksBorrowed INT DEFAULT 0 CHECK (TotalBooksBorrowed >= 0)
);

-- Table: BorrowRecords
CREATE TABLE BorrowRecords (
    BorrowID CHAR(6) PRIMARY KEY CHECK (BorrowID LIKE 'BR____'),
    BorrowDate DATE DEFAULT GETDATE(),
    MemberID CHAR(5) FOREIGN KEY REFERENCES Members(MemberID),
    BookID CHAR(5) FOREIGN KEY REFERENCES Books(BookID),
```

```
    QuantityBorrowed INT CHECK (QuantityBorrowed > 0)
);

-- Insert sample data into Books
INSERT INTO Books VALUES
('B0001', 'Book A', 'Author A', 'Fiction', 250.50, 10),
('B0002', 'Book B', 'Author B', 'Science', 300.00, 5),
('B0003', 'Book C', 'Author C', 'History', 150.00, 8);

-- Insert sample data into Members
INSERT INTO Members VALUES
('M0001', 'Alice', 'Smith', 'New York', 0),
('M0002', 'Bob', 'Johnson', 'Dhaka', 0),
('M0003', 'Charlie', 'Brown', 'Unknown', 0);

CREATE PROCEDURE AddBorrowRecord
    @BorrowID CHAR(6),
    @MemberID CHAR(5),
    @BookID CHAR(5),
    @Quantity INT
AS
BEGIN
    DECLARE @AvailableCopies INT;

    -- Check availability
    SELECT @AvailableCopies = CopiesInStock FROM Books WHERE BookID = @BookID;

    IF @AvailableCopies >= @Quantity
```

```
    BEGIN
        -- Insert the borrow record
        INSERT INTO BorrowRecords (BorrowID, MemberID, BookID, QuantityBorrowed)
        VALUES (@BorrowID, @MemberID, @BookID, @Quantity);

        -- Update Books table
        UPDATE Books
        SET CopiesInStock = CopiesInStock - @Quantity
        WHERE BookID = @BookID;

        -- Update Members table
        UPDATE Members
        SET TotalBooksBorrowed = TotalBooksBorrowed + @Quantity
        WHERE MemberID = @MemberID;

        PRINT 'Borrow record added successfully!';
    END
    ELSE
    BEGIN
        PRINT 'Not enough copies available.';
    END
END;

CREATE TRIGGER UpdateStockOnBorrow
ON BorrowRecords
AFTER INSERT
AS
BEGIN
```

```
    DECLARE @BookID CHAR(5), @Quantity INT;

    SELECT @BookID = BookID, @Quantity = QuantityBorrowed
    FROM INSERTED;

    -- Reduce stock in Books table
    UPDATE Books
    SET CopiesInStock = CopiesInStock - @Quantity
    WHERE BookID = @BookID;

    PRINT 'Stock updated!';
END;
```

2

```
CREATE DATABASE ECommerce;
USE ECommerce;

-- Table: Products
CREATE TABLE Products (
    ProductID CHAR(5) PRIMARY KEY CHECK (ProductID LIKE 'PR___'),
    ProductName VARCHAR(100) NOT NULL,
    Category VARCHAR(50) CHECK (Category IN ('Electronics', 'Clothing', 'Home Appliances')),
    Price DECIMAL(10, 2) CHECK (Price >= 0),
    StockQuantity INT CHECK (StockQuantity >= 0)
);

-- Table: Customers
```

```
CREATE TABLE Customers (
    CustomerID CHAR(5) PRIMARY KEY CHECK (CustomerID LIKE 'CU___'),
    FullName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Address VARCHAR(200)
);

-- Table: Orders
CREATE TABLE Orders (
    OrderID CHAR(5) PRIMARY KEY CHECK (OrderID LIKE 'OR___'),
    OrderDate DATE DEFAULT GETDATE(),
    CustomerID CHAR(5) FOREIGN KEY REFERENCES Customers(CustomerID),
    ProductID CHAR(5) FOREIGN KEY REFERENCES Products(ProductID),
    QuantityOrdered INT CHECK (QuantityOrdered > 0)
);

-- Insert sample data into Products
INSERT INTO Products VALUES
('PR001', 'Laptop', 'Electronics', 800.00, 15),
('PR002', 'Shirt', 'Clothing', 20.00, 50),
('PR003', 'Microwave', 'Home Appliances', 100.00, 10);

-- Insert sample data into Customers
INSERT INTO Customers VALUES
('CU001', 'Alice Johnson', 'alice@example.com', '123 Street A'),
('CU002', 'Bob Smith', 'bob@example.com', '456 Street B');
```

```
CREATE PROCEDURE AddOrder
    @OrderID CHAR(5),
    @CustomerID CHAR(5),
    @ProductID CHAR(5),
    @Quantity INT
AS
BEGIN
    DECLARE @Stock INT;

    -- Check stock
    SELECT @Stock = StockQuantity FROM Products WHERE ProductID = @ProductID;

    IF @Stock >= @Quantity
    BEGIN
        -- Insert the order
        INSERT INTO Orders (OrderID, CustomerID, ProductID, QuantityOrdered)
        VALUES (@OrderID, @CustomerID, @ProductID, @Quantity);

        -- Update stock
        UPDATE Products
        SET StockQuantity = StockQuantity - @Quantity
        WHERE ProductID = @ProductID;

        PRINT 'Order placed successfully!';
    END
    ELSE
    BEGIN
        PRINT 'Not enough stock available.';
```

```
    END
END;

CREATE TRIGGER RestockAlert
ON Orders
AFTER INSERT
AS
BEGIN
    DECLARE @ProductID CHAR(5), @NewStock INT;

    SELECT @ProductID = ProductID FROM INSERTED;

    -- Check stock level
    SELECT @NewStock = StockQuantity FROM Products WHERE ProductID = @ProductID;

    IF @NewStock < 10
    BEGIN
        PRINT 'Restock Alert: Stock for Product ' + @ProductID + ' is below 10!';
    END
END;
```

3

```
CREATE DATABASE School;
USE School;

-- Table: Students
CREATE TABLE Students (
```

```
    StudentID CHAR(4) PRIMARY KEY CHECK (StudentID LIKE 'S___'),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Grade INT CHECK (Grade BETWEEN 1 AND 12),
    City VARCHAR(50) DEFAULT 'Not Specified'
);

-- Table: Subjects
CREATE TABLE Subjects (
    SubjectID CHAR(4) PRIMARY KEY CHECK (SubjectID LIKE 'SU___'),
    SubjectName VARCHAR(100) NOT NULL,
    Teacher VARCHAR(100) NOT NULL
);

-- Table: Enrollments
CREATE TABLE Enrollments (
    EnrollmentID CHAR(5) PRIMARY KEY CHECK (EnrollmentID LIKE 'E____'),
    EnrollmentDate DATE DEFAULT GETDATE(),
    StudentID CHAR(4) FOREIGN KEY REFERENCES Students(StudentID),
    SubjectID CHAR(4) FOREIGN KEY REFERENCES Subjects(SubjectID)
);

-- Insert sample data into Students
INSERT INTO Students VALUES
('S001', 'Alice', 'Smith', 5, 'New York'),
('S002', 'Bob', 'Johnson', 7, 'Los Angeles'),
('S003', 'Charlie', 'Brown', 10, 'Chicago');
```

```
-- Insert sample data into Subjects
INSERT INTO Subjects VALUES
('SU01', 'Mathematics', 'Mr. Johnson'),
('SU02', 'Science', 'Ms. Lee'),
('SU03', 'History', 'Mr. Smith');

CREATE PROCEDURE AddEnrollment
    @EnrollmentID CHAR(5),
    @StudentID CHAR(4),
    @SubjectID CHAR(4)
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM Enrollments
        WHERE StudentID = @StudentID AND SubjectID = @SubjectID
    )
    BEGIN
        PRINT 'Error: The student is already enrolled in this subject.';
    END
    ELSE
    BEGIN
        INSERT INTO Enrollments (EnrollmentID, StudentID, SubjectID)
        VALUES (@EnrollmentID, @StudentID, @SubjectID);
        PRINT 'Enrollment successful.';
    END
END;

CREATE TABLE EnrollmentLog (
```

```
    LogID INT IDENTITY PRIMARY KEY,
    EnrollmentID CHAR(5),
    StudentID CHAR(4),
    SubjectID CHAR(4),
    LogDate DATETIME DEFAULT GETDATE()
);

CREATE TRIGGER LogEnrollment
ON Enrollments
AFTER INSERT
AS
BEGIN
    INSERT INTO EnrollmentLog (EnrollmentID, StudentID, SubjectID)
    SELECT EnrollmentID, StudentID, SubjectID
    FROM INSERTED;

    PRINT 'Enrollment log updated.';
END;
```

4

```
CREATE DATABASE Hospital;
USE Hospital;

-- Table: Doctors
CREATE TABLE Doctors (
    DoctorID CHAR(4) PRIMARY KEY CHECK (DoctorID LIKE 'D___'),
    Name VARCHAR(100) NOT NULL,
```

```
    Specialty VARCHAR(50) NOT NULL,
    AvailableSlots INT CHECK (AvailableSlots >= 0)
);

-- Table: Patients
CREATE TABLE Patients (
    PatientID CHAR(4) PRIMARY KEY CHECK (PatientID LIKE 'P___'),
    Name VARCHAR(100) NOT NULL,
    Address VARCHAR(200),
    PhoneNumber VARCHAR(15) UNIQUE
);

-- Table: Appointments
CREATE TABLE Appointments (
    AppointmentID CHAR(5) PRIMARY KEY CHECK (AppointmentID LIKE 'A____'),
    AppointmentDate DATE NOT NULL,
    DoctorID CHAR(4) FOREIGN KEY REFERENCES Doctors(DoctorID),
    PatientID CHAR(4) FOREIGN KEY REFERENCES Patients(PatientID)
);

-- Insert sample data into Doctors
INSERT INTO Doctors VALUES
('D001', 'Dr. Smith', 'Cardiologist', 5),
('D002', 'Dr. Lee', 'Neurologist', 3),
('D003', 'Dr. Brown', 'Dermatologist', 2);

-- Insert sample data into Patients
INSERT INTO Patients VALUES
```

```
('P001', 'Alice Johnson', '123 Main Street', '1234567890'),
('P002', 'Bob Smith', '456 Oak Avenue', '9876543210'),
('P003', 'Charlie Brown', '789 Pine Road', '5554443333');

CREATE PROCEDURE BookAppointment
    @AppointmentID CHAR(5),
    @AppointmentDate DATE,
    @DoctorID CHAR(4),
    @PatientID CHAR(4)
AS
BEGIN
    DECLARE @Slots INT;

    -- Check available slots
    SELECT @Slots = AvailableSlots FROM Doctors WHERE DoctorID = @DoctorID;

    IF @Slots > 0
    BEGIN
        -- Insert appointment
        INSERT INTO Appointments (AppointmentID, AppointmentDate, DoctorID, PatientID)
        VALUES (@AppointmentID, @AppointmentDate, @DoctorID, @PatientID);

        -- Update available slots
        UPDATE Doctors
        SET AvailableSlots = AvailableSlots - 1
        WHERE DoctorID = @DoctorID;

        PRINT 'Appointment booked successfully.';
```

```
    END
    ELSE
    BEGIN
        PRINT 'Error: No available slots for this doctor.';
    END
END;

CREATE TRIGGER NotifySlots
ON Appointments
AFTER INSERT
AS
BEGIN
    DECLARE @DoctorID CHAR(4), @Slots INT;

    SELECT @DoctorID = DoctorID FROM INSERTED;

    -- Check remaining slots
    SELECT @Slots = AvailableSlots FROM Doctors WHERE DoctorID = @DoctorID;

    IF @Slots = 0
    BEGIN
        PRINT 'Notification: Doctor ' + @DoctorID + ' has no available slots remaining.';
    END
END;
```

5 Joint

```
CREATE DATABASE CompanyDB;

USE CompanyDB;

-- Table: Departments

CREATE TABLE Departments (

   DepartmentID CHAR(4) PRIMARY KEY CHECK (DepartmentID LIKE 'D___'),

   DepartmentName VARCHAR(50) NOT NULL,

   Location VARCHAR(50) NOT NULL

);


-- Table: Employees

CREATE TABLE Employees (

   EmployeeID CHAR(4) PRIMARY KEY CHECK (EmployeeID LIKE 'E___'),

   FirstName VARCHAR(50) NOT NULL,

   LastName VARCHAR(50) NOT NULL,

   DepartmentID CHAR(4) FOREIGN KEY REFERENCES Departments(DepartmentID)

);


-- Insert data into Departments

INSERT INTO Departments VALUES

('D001', 'Human Resources', 'New York'),

('D002', 'IT', 'San Francisco'),

('D003', 'Marketing', 'Chicago');


-- Insert data into Employees

INSERT INTO Employees VALUES

('E001', 'Alice', 'Johnson', 'D001'),
```

```
('E002', 'Bob', 'Smith', 'D002'),
('E003', 'Charlie', 'Brown', 'D003');

SELECT
    Employees.EmployeeID,
    Employees.FirstName,
    Employees.LastName,
    Departments.DepartmentName,
    Departments.Location
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```