ASSIGNMENT – 4

CSA0593 – DBMS

NAME : S.DEEPIKA

REG NO : 192324269

1.Create a database to manage books, authors, genres, and user reading lists.

* Model tables for books, authors, genres, and reading lists.
* Write stored procedures for adding books to reading lists and updating genre preferences.
* Implement triggers to automatically update book recommendation scores based on user ratings and completion.
* Write SQL queries to analyze popular books, genre trends, and user reading behavior.

ANSWER :

CONCEPTUAL ER DIAGRAM:

1.Entities: Books(BookID, Title, AuthorID, GenreID), Authors(AuthorID, Name), Genres(GenreID, Name), Users(UserID, Name), ReadingLists(ListID, UserID, BookID).

2. Relationships: Authors 1:N Books, Genres 1:N Books, Users N:M Books through ReadingLists.

3. Keys: Books(AuthorID, GenreID foreign keys), ReadingLists(UserID, BookID foreign keys).

CODE:

CREATE DATABASE BookManagement;

USE BookManagement;

CREATE TABLE Authors (

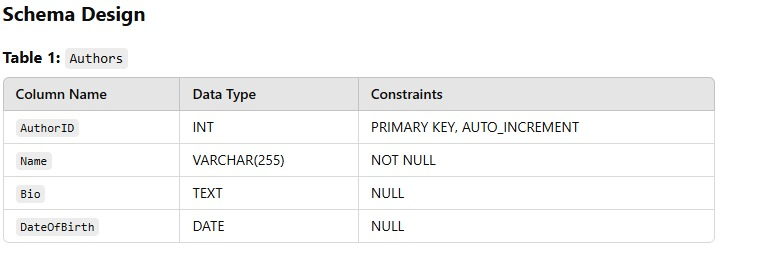
AuthorID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Bio TEXT,

DateOfBirth DATE

);



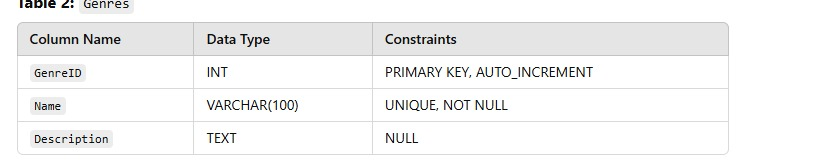
CREATE TABLE Genres (

GenreID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100) UNIQUE NOT NULL,

Description TEXT

);



CREATE TABLE Books (

BookID INT AUTO\_INCREMENT PRIMARY KEY,

Title VARCHAR(255) NOT NULL,

AuthorID INT NOT NULL,

GenreID INT NOT NULL,

PublicationDate DATE,

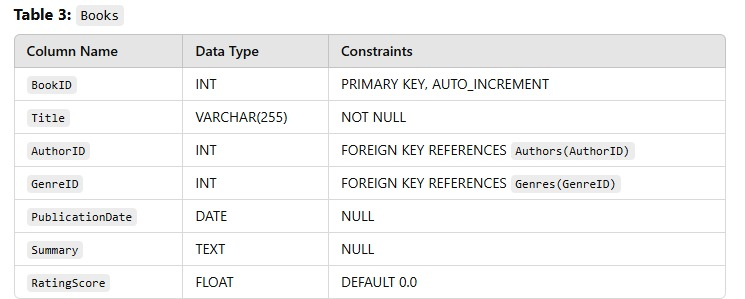
Summary TEXT,

RatingScore FLOAT DEFAULT 0.0,

FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID),

FOREIGN KEY (GenreID) REFERENCES Genres(GenreID)

);



CREATE TABLE Users (

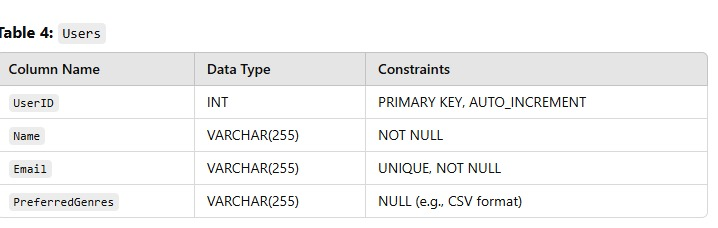
UserID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

PreferredGenres VARCHAR(255)

);



CREATE TABLE ReadingLists (

ListID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT NOT NULL,

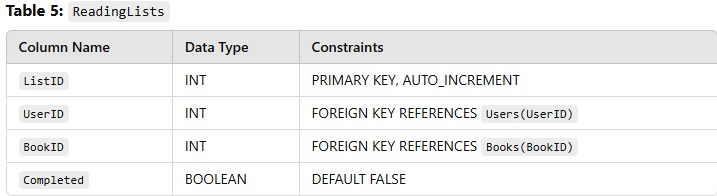
BookID INT NOT NULL,

Completed BOOLEAN DEFAULT FALSE,

FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (BookID) REFERENCES Books(BookID)

);



DELIMITER //

CREATE PROCEDURE AddBookToReadingList (

IN p\_UserID INT,

IN p\_BookID INT

)

BEGIN

IF NOT EXISTS (SELECT 1 FROM ReadingLists WHERE UserID = p\_UserID AND BookID = p\_BookID) THEN

INSERT INTO ReadingLists (UserID, BookID) VALUES (p\_UserID, p\_BookID);

END IF;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE UpdateGenrePreferences (

IN p\_UserID INT,

IN p\_GenreList VARCHAR(255)

)

BEGIN

UPDATE Users

SET PreferredGenres = p\_GenreList

WHERE UserID = p\_UserID;

END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER UpdateBookRatingScore

AFTER UPDATE ON ReadingLists

FOR EACH ROW

BEGIN

IF NEW.Completed = TRUE THEN

UPDATE Books

SET RatingScore = RatingScore + 1

WHERE BookID = NEW.BookID;

END IF;

END //

DELIMITER ;

-- Analyze Popular Books

SELECT Title, RatingScore

FROM Books

ORDER BY RatingScore DESC

LIMIT 10;

-- Analyze Genre Trends

SELECT g.Name AS Genre, COUNT(r.BookID) AS BookCount

FROM Genres g

JOIN Books b ON g.GenreID = b.GenreID

JOIN ReadingLists r ON b.BookID = r.BookID

GROUP BY g.Name

ORDER BY BookCount DESC;

-- Analyze User Reading Behavior

SELECT u.Name AS User, COUNT(r.BookID) AS CompletedBooks

FROM Users u

JOIN ReadingLists r ON u.UserID = r.UserID

WHERE r.Completed = TRUE

GROUP BY u.UserID

ORDER BY CompletedBooks DESC;

CONCLUSION

This SQL implementation creates a robust database to manage books, authors, genres, users, and reading lists. Stored procedures streamline operations, triggers automate updates like recommendation scores, and analytical queries provide insights into trends and user behavior. The design ensures efficiency, scalability, and data-driven decision-making.