Task 4: Build a Power BI Dashboard for a School Library Management System.

1. SQL Script (Entitite) :

```
show databases;
create database task4;
use task4;
-- Author Table
CREATE TABLE Author (
    AuthorID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR (50),
    LastName VARCHAR (50),
    Biography TEXT
);


-- Genre Table
CREATE TABLE Genre (
    GenreID INT PRIMARY KEY AUTO_INCREMENT,
    GenreName VARCHAR (50)
);

-- Book Table
CREATE TABLE Book (
    ISBN VARCHAR (13) PRIMARY KEY,
    Title VARCHAR (100),
    AuthorID INT,
    GenreID INT,
    PublishedYear YEAR,
    AvailableCopies INT,
```

```
    TotalCopies INT,
    ShelfLocation VARCHAR (20),
    BookStatus ENUM ('Available', 'Checked Out') DEFAULT 'Available',
    FOREIGN KEY (AuthorID) REFERENCES Author (AuthorID),
    FOREIGN KEY (GenreID) REFERENCES Genre (GenreID)
);

-- Student Table
CREATE TABLE Student (
    StudentID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR (50),
    LastName VARCHAR (50),
    Grade VARCHAR (10),
    ContactInfo VARCHAR (100)
);

-- Checkout Table
CREATE TABLE Checkout (
    CheckoutID INT PRIMARY KEY AUTO_INCREMENT,
    StudentID INT,
    ISBN VARCHAR (13),
    CheckoutDate DATE,
    ReturnDate DATE,
    Status ENUM ('Checked Out', 'Returned') DEFAULT 'Checked Out',
    FOREIGN KEY (StudentID) REFERENCES Student (StudentID),
    FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
);

-- Reservation Table
CREATE TABLE Reservation (
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    StudentID INT,
    ISBN VARCHAR (13),
    ReservationDate DATE,
    PickupDeadline DATE,
    Status ENUM ('Pending', 'Expired', 'Fulfilled') DEFAULT 'Pending',
    FOREIGN KEY (StudentID) REFERENCES Student (StudentID),
    FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
);

-- Fine Table
CREATE TABLE Fine (
    FineID INT PRIMARY KEY AUTO_INCREMENT,
    StudentID INT,
    Amount DECIMAL (6,2),
    IssuedDate DATE,
    Status ENUM ('Paid', 'Unpaid') DEFAULT 'Unpaid',
    FOREIGN KEY (StudentID) REFERENCES Student (StudentID)
);

-- Review Table
CREATE TABLE Review (
    ReviewID INT PRIMARY KEY AUTO_INCREMENT,
    ISBN VARCHAR (13),
    StudentID INT,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    Comment TEXT,
    Date DATE,
    FOREIGN KEY (ISBN) REFERENCES Book (ISBN),
    FOREIGN KEY (StudentID) REFERENCES Student (StudentID));
```

-- Insert into Author

INSERT INTO Author (FirstName, LastName, Biography) VALUES

('J.K.', 'Rowling', 'British author, best known for the Harry Potter series'),

('George', 'Orwell', 'English novelist and essayist'),

('Jane', 'Austen', 'English novelist known for romantic fiction');

-- Insert into Genre

INSERT INTO Genre (GenreName) VALUES

('Fantasy'),

('Science Fiction'),

('Romance'),

('Mystery');

-- **Insert into Book**

INSERT INTO Book (ISBN, Title, AuthorID, GenreID, PublishedYear, AvailableCopies, TotalCopies, ShelfLocation, BookStatus) VALUES

('9780141439518', 'Pride and Prejudice', 3, 3, 1901, 1, 3, 'C3', 'Available'),

(9780141439535, 'Pride and Prejudice', 3, 3, 1901, 1, 3, 'C3', 'Available'),

('9780439139601', 'Harry Potter and the Goblet of Fire', 1, 1, 2000, 3, 5, 'A1', 'Available'),

('9780451524935', 'kings' daughter', 2, 2, 1949, 2, 4, 'B2', 'Checked Out'),

('9780553573428', 'A Game of Thrones', 2, 3, 1996, 5, 6, 'A2', 'Checked Out');

-- Insert into Student

INSERT INTO Student (FirstName, LastName, Grade, ContactInfo) VALUES

('Alice', 'Johnson', '10', 'alice.johnson@email.com'),

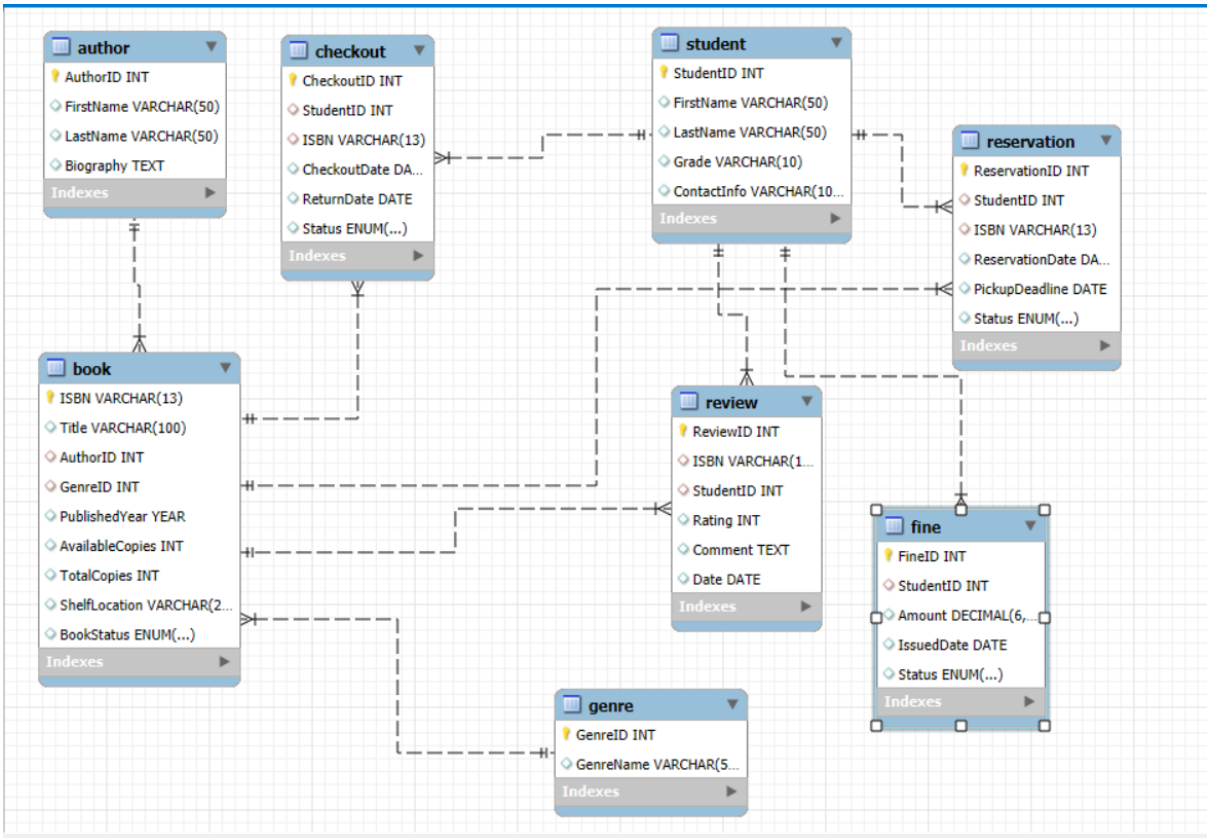
('Bob', 'Smith', '11', 'bob.smith@email.com'),

('Clara', 'Lee', '12', 'clara.lee@email.com');

-- Insert into Checkout

```
INSERT INTO Checkout (StudentID, ISBN, CheckoutDate, ReturnDate, Status) VALUES
(1, '9780451524935', '2025-07-10', '2025-07-20', 'Returned'),
(2, '9780439139601', '2025-07-15', '2025-07-25', 'Checked Out');

-- Insert into Reservation
INSERT INTO Reservation (StudentID, ISBN, ReservationDate, PickupDeadline, Status) VALUES
(3, '9780451524935', '2025-07-16', '2025-07-20', 'Pending'),
(1, '9780141439518', '2025-07-12', '2025-07-15', 'Fulfilled');

-- Insert into Fine
INSERT INTO Fine (StudentID, Amount, IssuedDate, Status) VALUES
(1, 10.00, '2025-07-21', 'Paid'),
(2, 5.00, '2025-07-22', 'Unpaid');

-- Insert into Review
INSERT INTO Review (ISBN, StudentID, Rating, Comment, Date) VALUES
('9780439139601', 2, 5, 'Amazing book, loved it!', '2025-07-18'),
('9780141439518', 1, 4, 'A classic romance story', '2025-07-19');

select*from Author;
select* from book;
select* from student;
select*from Checkout;
select*from Reservation;
select*from Fine;
select*from Review;
show tables;
```

2. ERD (Entity Relationship Diagram)



3. Design Document

Design Document (Report)

a. Design Choices

Tables Created and Purpose:

• **Author:** Stores information about book authors. Helps normalize the data and avoid repeating author names in every book entry.

• **Genre:** Categorizes books into genres (e.g., Fiction, Science, History) for better filtering and reports.

• **Book:** Main entity representing all library books. Includes ISBN, Title, AuthorID, GenreID, availability, and shelf details.

• **Borrower:** Holds student/user details. Essential for issuing and tracking books.

• **Issue:** Records each instance when a book is issued to a student, with issue date and due date.

• **Return**: Stores return date and any fine amount, connected to the Issue table.

• **Fine**: Maintains fine history separately, allowing detailed reporting (reason, amount, date).

b. Data Type Choices:

• **INT:** Used for primary keys (e.g., BookID, BorrowerID) for simplicity, indexing, and

auto-increment.

• **VARCHAR**: Used for textual fields like Name, Title, GenreName, ContactNumber, and Email to allow flexibility in input.

• **DATE:** Chosen for IssueDate, DueDate, ReturnDate, and FineDate for accurate date calculations.

• **DECIMAL (6,2):** Used for FineAmount to handle monetary values properly.

b. Assumptions

• Each student can borrow up to 3 books at a time.

• Books must be returned within 14 days of issue.

• If a book is returned late, a fine of ₹2 per day is applied.

• Only students registered in the system can borrow books.

• A book is marked "Unavailable" if all copies are issued.

c. Additional Features

1. Fine Calculation Logic:

o Return table captures fine amount.

o System checks if the return date is beyond the due date.

o Fine = Days overdue * ₹2

2. Book Availability Check:

o Before issuing a book, system checks AvailableCopies > 0.

3. Overdue Alerts:

o Queries can highlight students who haven't returned books beyond due date.

o Power BI can show these as charts or alerts.

4. Popular Books/Genres Tracking:

o Use Issue table to count how many times a book or genre has been issued.

5. Student Borrowing History:

o Join Borrower, Issue, Return to track borrowing habits per student.

d. Enhancements and Their Benefits

1. For Librarians: - Can quickly find overdue books and apply fines. - Real-time insights into which books are most or least used. - Reduces time spent tracking returns manually.

2. For Students: - Know which books are available before visiting. - Avoid fines by getting

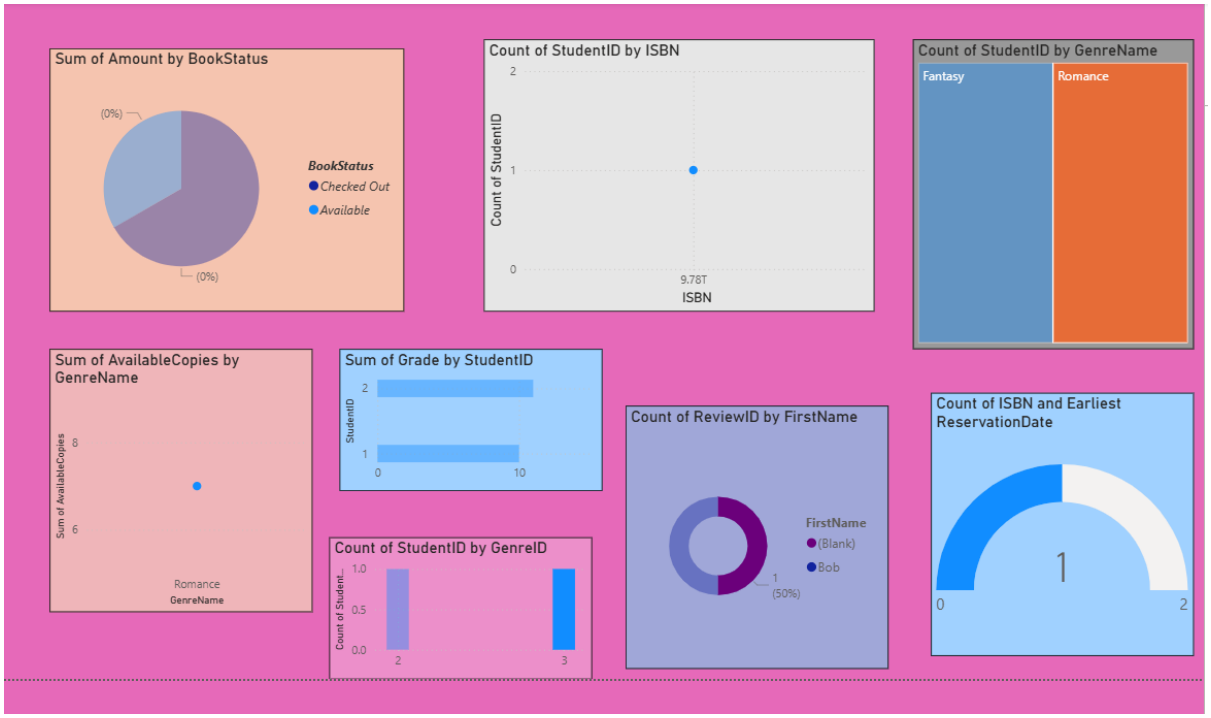
alerts or viewing due dates. - See their own borrowing history.

3. For Administration: - Analyse patterns of overdue or lost books. - Promote reading by seeing which genres students prefer. - Make better decisions on book purchasing and stocking.

Prepared by: Divya Shree

Tool Used: MySQL Workbench, Power BI.

4. Power BI Reports



5. Insights and Recommendations (Include in Design Document)

****