MARMARA
ÜNİVERSİTESİ

DATABASE MANAGEMENT
LIBRARY MANAGEMENT SYSTEM

Assoc. Prof. Selçuk KIRAN

İsmail ERDEN / 138723014

Zeliha MUTLU / 138724011

Nusret ORTAÇ / 138723016

Taibenur YAVUZ / 138723007

Zeliha MUTLU / 138724011

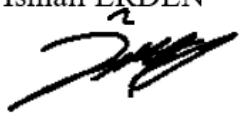
Nusret ORTAÇ / 138723016

Taibenur YAVUZ / 138723007

İsmail ERDEN / 138723014

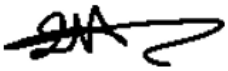
I swear, in the name of my honor and all the values I believed in, that I

did not copy and paste anything from anywhere during this project.

İsmail ERDEN 

Taibenur YAVUZ 

Nusret ORTAÇ 

Zeliha MUTLU 

1. Brief Description of the System

The Library Management System developed within this project is designed to manage the daily operations of a library in an efficient, organized, and secure manner.
The system tracks and records all processes such as book borrowing, fines, and donations — ensuring that accurate information about members, librarians, and books is maintained.

Through this system, library staff can easily monitor the status of books, manage member transactions, track fines, and control donations.
Additionally, the system ensures that all data — such as book information, author details, borrowing history, fines, and donation records — are securely stored and can be accessed when needed.
This structure facilitates communication between librarians and members while reducing manual recording errors.

Our Aim in Designing this Project

1. To ensure that all library transactions are systematically and efficiently recorded.
2. To make book borrowing and fine processes more accurate and traceable.
3. To establish a clear penalty system that restricts borrowing for members with overdue fines.
4. To create a reliable and user-friendly system for both librarians and members.

1.1 Scope Definition

The Library Management System will be accessible to both librarians and members. Members will be able to borrow books, make donations, and keep track of their fines and restrictions through the system.
Librarians will be able to manage member records, approve transactions, and process borrowing and fine-related operations.

The system will operate on a relational database structure, where all entities — such as books, members, authors, and donations — are interconnected, and all data is securely stored within the database.

1.2 Determining System Requirements

The system must be capable of performing the following functions:

- Recording all book, author, and category information.
- Tracking which books are borrowed, by whom, and when they are due for return.
- Automatically generating fines for overdue books and restricting members from borrowing new books for 30 days.
- Recording donations made by members and ensuring that they are approved by librarians.
- Ensuring that all data is securely stored and accessible only to authorized users.

The Requirements Are as Follows

1. A member can borrow multiple books; however, each book can be borrowed by only one member at a time.
2. Librarians have the authority to approve and manage all borrowing transactions.
3. The system automatically generates a fine for overdue books.
4. A fine results in a 30-day borrowing restriction for the related member.
5. Members can make donations, which must be approved by librarians.
6. Each book belongs to only one category and is located on a specific shelf.
7. Each book is written by one author.
8. Borrowing transactions clearly define the organized relationship between members, books, and librarians.
9. All records related to borrowing, fines, and donations are securely and traceably stored in the system.
10. Donations made are related to both members and related books and are recorded within the system.

1.3 Entities

- Member

Member will be the one who borrows or donates books in the library system. He should be a registered member of the system and the system gives member ID to each user. Member entity has member ID, first name, last name, email, address, phone.

- Librarian

Librarian represents the staff member who manages all the library operations such as adding or removing books, confirming donations, and updating borrow records. Librarian entity has librarian ID, first name, last name, email, phone, hire date.

- Book

Book entity will have the list of all books available in the library. Each book is connected to an author, category, and shelf location. Book entity has book ID, title, author ID, category ID, book ISBN, book publisher, book publish year, book count, book shelf ID, book status.

- Category

Category defines the type or classification of books in the library. It helps users to search and filter books based on their interests. Category entity has category ID, name.

- Author

Author entity stores information about writers whose books exist in the library system. Each author can have multiple books. Author entity has author ID, first name, last name, nationality.

- Borrow

Borrow entity records the information about borrowing transactions between members and books. It helps track when a book is borrowed and when it is returned. Borrow entity has borrow ID, member ID, book ID, borrow date, due date, return date, status.

- Fine

Fine entity represents the charge that is applied to members who return books late or damage them. It is linked with the borrow record. Fine entity has fine ID, borrow ID, fine reason, fine issue date, fine status.

- Shelf Location

Shelf Location entity shows the physical position of a book in the library. It helps both members and librarians to find and organize books easily. Shelf Location entity has shelf ID, section name, floor number, shelf row number.

- Donation

Donation entity represents the books or materials donated by members to the library. The librarian checks and approves these donations before adding them to the system. Donation entity has donation ID, member ID, book ID, donation date, librarian ID.

- Borrow_Book (Composite Entities)

  Borrow_Book represents the detailed relationship between the Borrow and Book entities. It records which specific books are included in each borrowing transaction. Through this entity, the system can track multiple books borrowed under a single borrow record and maintain book-level details such as due date, return date, and borrowing status. Borrow_Book entity also links fines applied to specific books when they are returned late or damaged. The entity has Borrow_ID, Book_ID, Fine_ID, Due_Date, Return_Date, and Item_Status.

2. Components of the Entity Relationship Model

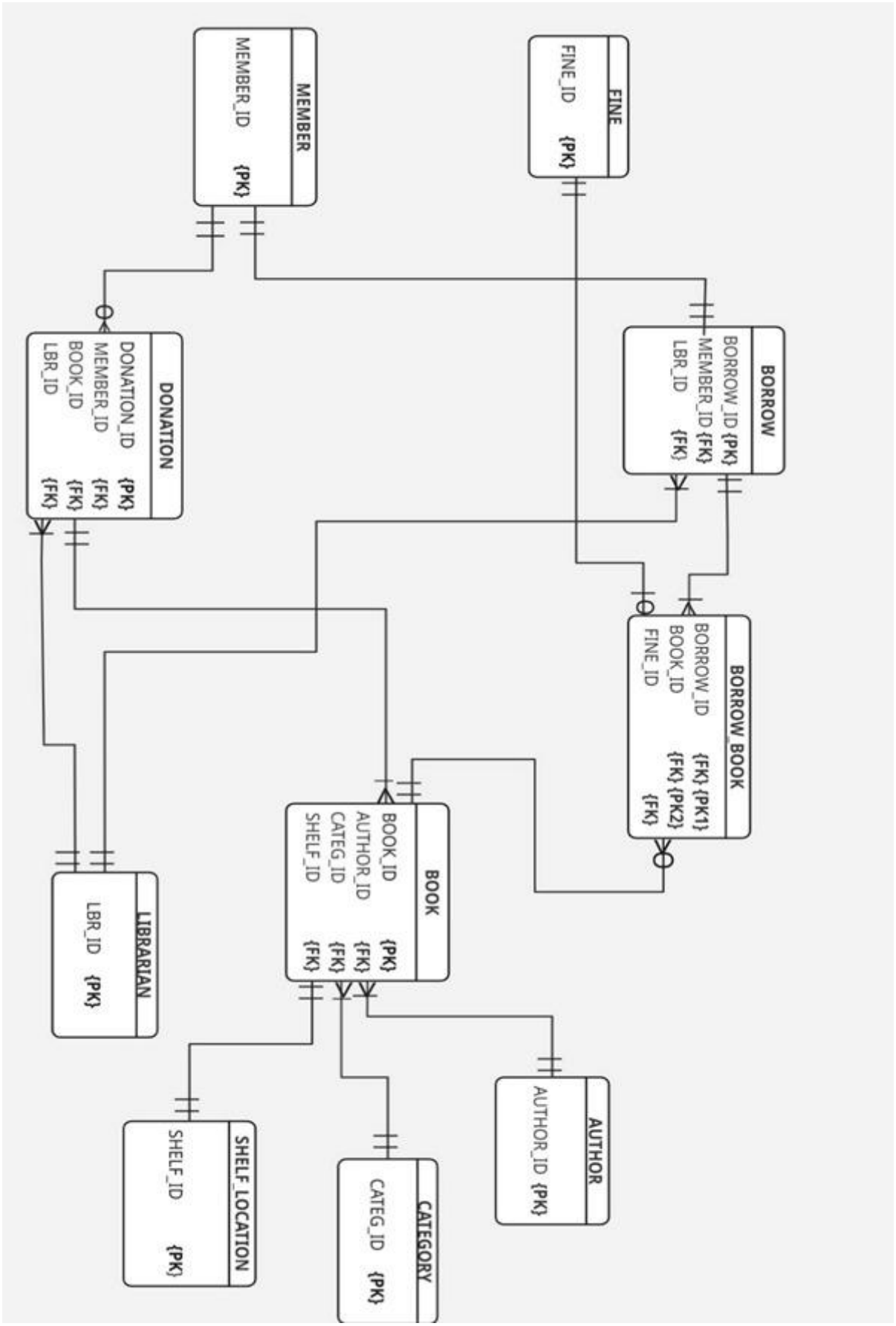
ENTITY	**RELATIONSHIP**	**CONNECTIVITY**	**ENTITY**
CATEGORY	Classifies	1 - *	BOOK
SHELF_LOCATION	Holds	1 - *	BOOK
AUTHOR	Writes	1 - *	BOOK
MEMBER	Makes	1 - *	BORROW
LIBRARIAN	Processes	1 - *	BORROW
BOOK	Included in	* - *	BORROW(via BORROW_BOOK)
BORROW	Contains	1 - *	BORROW_BOOK
BOOK	Is part of	1 - *	BORROW_BOOK
BORROW_BOOK	Generates	0 - 1	FINE
MEMBER	Donates	1 - *	DONATION
LIBRARIAN	Approves	1 - *	DONATION
BOOK	Received in	1 - *	DONATION

3. Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Domain	Required	PK or FK	FK Referenced Table
Member	MEMBER_ID	Unique identification number assigned to each library member.	INT	#######	0–9999999	Y	PK	
	MEMBER_FN	Member's first name.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	MEMBER_LN	Member's last name.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	MEMBER_MAIL	Member's email address.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	MEMBER_ADDR	Member's home or contact address.	VARCHAR(150	Xxxxxxxxxxxxxxxx	NA	Y		
	MEMBER_PHONE	Member's phone number.	CHAR(10)	999-999-9999	NA	Y		
Librarian	LBR_ID	Unique identification number assigned to each librarian.	INT	#######	0–9999999	Y	PK	
	LBR_FN	Librarian's first name.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	LBR_LN	Librarian's last name.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	LBR_MAIL	Librarian's email address.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	LBR_PHONE	Librarian's contact phone number.	CHAR(10)	999-999-9999	NA	Y		
	LBR_H_DATE	Librarian's hire date (the date when they started working).	DATE	dd-mon-yyyy	NA	Y		
Book	BOOK_ID	Unique identification number for each book.	INT	#######	0–9999999	Y	PK	
	BOOK_TITLE	Title of the book.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	AUTHOR_ID	Identification number of the book's author.	INT	#######	0–9999999	Y	FK	Author
	CATEG_ID	Identification number of the category the book belongs to.	INT	#######	0–9999999	Y	FK	Category
	BOOK_ISBN	International Standard Book Number (ISBN).	VARCHAR(20)	Xxxxxxxxxxxxxxxx	NA	Y		
	BOOK_PUBL	Publisher name of the book.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	BOOK_PUBL_YEAR	Year of publication of the book.	SMALLINT	####	0–9999	Y		
	BOOK_COUNT	Total number of available copies of the book.	SMALLINT	##	0–99	Y		
	BOOK_SHELF_ID	Shelf identification number where the book is located.	INT	#######	0–9999999	Y	FK	Shelf Location
	BOOK_STATUS	Current status of the book (Available / Borrowed / Reserved).	CHAR(10)	Xxxxxxxxx	NA	Y		
Category	CATEG_ID	Unique identification number assigned to each category.	INT	#######	0–9999999	Y	PK	
	CATEG_NAME	Name of the book category (e.g., Science, Literature, History).	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
Author	AUTHER_ID	Unique identification number assigned to each author.	INT	#######	0–9999999	Y	PK	
	AUTHER_FN	Author's first name.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	AUTHER_LN	Author's last name.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	AUTHER_NATİON	Author's nationality.	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
Borrow	BORROW_ID	Unique identification number for each borrow transaction.	INT	#######	0–9999999	Y	PK	
	MEMBER_ID	Identification number of the member who borrowed the book.	INT	#######	0–9999999	Y	FK	Member
	BOOK_ID	Identification number of the borrowed book.	INT	#######	0–9999999	Y		
	BORROW_DATE	Date when the book was borrowed.	DATE	dd-mon-yyyy	NA	Y		
	BORROW_DUEDATE	Due date for returning the borrowed book.	DATE	dd-mon-yyyy	NA	Y		
	BORROW_R_DATE	Actual date when the book was returned.	DATE	dd-mon-yyyy	NA	Y		
	BORROW_STATUS	Current status of the borrow transaction (Ongoing / Returned / Overdue).	CHAR(10)	Xxxxxxxxx	NA	Y		
	LBR_ID	Identification number of the librarian who processed the transaction.	INT	#######	0–9999999	Y	FK	Librarian
Fine	FINE_ID	Unique identification number for each fine record.	INT	#######	0–9999999	Y	PK	
	BORROW_ID	Identification number of the borrow transaction related to the fine.	INT	#######	0–9999999	Y		
	FINE_REASON	Reason for the fine (Late return, Lost book, Damaged book).	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	FINE_I_DATE	Date when the fine was issued.	DATE	dd-mon-yyyy	NA	Y		
	FINE_STATUS	Status of the fine (Active / Resolved / Waived).	CHAR(10)	Xxxxxxxxx	NA	Y		
Shelf Location	SHELF_ID	Unique identification number for each shelf.	INT	#######	0–9999999	Y	PK	
	SHELF_SECT_NAME	Name or section of the shelf (e.g., Literature Section, Science Corner).	VARCHAR(50)	Xxxxxxxxxxxxxxxx	NA	Y		
	SHELF_FLOOR	Floor number where the shelf is located.	SMALLINT	##	0–99	Y		
	SHELF_ROW_NUM	Row number or code of the shelf.	VARCHAR(10)	Xxxxxxxxx	NA	Y		
Donation	DONATION_ID	Unique identification number for each donation record.	INT	#######	0–9999999	Y	PK	
	MEMBER_ID	Identification number of the member who donated the book.	INT	#######	0–9999999	Y	FK	Member
	BOOK_ID	Identification number of the donated book.	INT	#######	0–9999999	Y	FK	Book
	DONATION_DATE	Date when the donation was made.	DATE	dd-mon-yyyy	NA	Y		
	LBR_ID	ID of librarian approving the donation.	INT	#######	0–9999999	Y	FK	Librarian
Borrow_Book (Composite Entities)	BORROW_ID	Identification number of the borrowing transaction	INT	#######	0–9999999	Y	FK (PK1)	Borrow
	BOOK_ID	Identification number of the borrowed book.	INT	#######	0–9999999	Y	FK (PK2)	Book
	FINE_ID	Identification number of the fine related to this borrowed item.	INT	#######	0–9999999	N	FK	Fine
	DUE_DATE	The date by which the borrowed book must be returned.	DATE	dd-mon-yyyy	NA	N		
	RETURN_DATE	The actual date when the book was returned.	DATE	dd-mon-yyyy	NA	Y		
	ITEM_STATUS	Indicates the current state of the borrowed item	CHAR(15)	Returned	NA	Y		

PK – Primary Key, unique record identifier.
FK – Foreign Key, references another table's primary key.
INT – Integer numeric value (whole number).
VARCHAR(n) – Variable-length text up to *n* characters.
CHAR(n) – Fixed-length text, always *n* characters.
NUMBER(8,2) – Numeric value with two decimals (e.g., 1234.56).
SMALLINT – Small integer value (0–99).
DATE – Date format dd-mon-yyyy.
TIME – Time format hh:mm.
NA – Not applicable / no restriction.

4. Entity Relationship Model



```sql
CREATE TABLE Category(

    Category_ID int PRIMARY KEY,
    Category_Name varchar(20)
);

CREATE TABLE Author (
    Author_ID int PRIMARY KEY,
    Author_FN varchar(20),
    Author_LN varchar(20),
    Author_Nation varchar(20)
);

CREATE TABLE ShelfLocation (
    Shelf_ID int PRIMARY KEY,
    Shelf_SectName varchar(20),
    Shelf_Floor smallint,
    Shelf_RowNum varchar(10)
);

CREATE TABLE Member(
    Member_ID int PRIMARY KEY,
    Member_FN varchar(15),
    Member_LN varchar(15),
    Member_MAIL varchar(30),
    Member_ADDR  varchar(60),
    Member_PHONE  char(10)
);

CREATE TABLE Librarian (
    LBR_ID int PRIMARY KEY,
    LBR_FN varchar(15),
    LBR_LN  varchar(15),
    LBR_MAIL varchar(30),
    LBR_PHONE char(10),
    LBR_H_DATE  date
);
CREATE TABLE Book (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(30) NOT NULL,
    Author_ID INT NOT NULL,
    Category_ID INT NOT NULL,
    ISBN VARCHAR(20) UNIQUE,
    Publisher VARCHAR(20),
    PublishYear SMALLINT,
    BookCount INT DEFAULT 1,
    Shelf_ID INT NOT NULL,
    BookStatus VARCHAR(20),

    CONSTRAINT fk_book_author FOREIGN KEY (Author_ID) REFERENCES
Author(Author_ID),
    CONSTRAINT fk_book_category FOREIGN KEY (Category_ID) REFERENCES
Category(Category_ID),
    CONSTRAINT fk_book_shelf FOREIGN KEY (Shelf_ID) REFERENCES
ShelfLocation(Shelf_ID)
);
```

```sql
CREATE TABLE Borrow (
    Borrow_ID int PRIMARY KEY,
    Borrow_Date date,
    Member_ID int NOT NULL,
    LBR_ID int NOT NULL,

    CONSTRAINT fk_borrow_member FOREIGN KEY (Member_ID) REFERENCES
Member(Member_ID),
    CONSTRAINT fk_borrow_librarian FOREIGN KEY(LBR_ID) REFERENCES
Librarian(LBR_ID)
);
CREATE TABLE BorrowBook(
    Borrow_ID int NOT NULL,
    Book_ID int NOT NULL,
    DueDate date,
    ReturnDate date,
    ItemStatus varchar(15),


    PRIMARY KEY(Borrow_ID, Book_ID),

    CONSTRAINT fk_bb_borrow FOREIGN KEY(Borrow_ID) REFERENCES
Borrow(Borrow_ID),
    CONSTRAINT fk_bb_book FOREIGN KEY(Book_ID) REFERENCES Book(Book_ID)
);
CREATE TABLE Fine(
    Fine_ID int PRIMARY KEY,
    Borrow_ID int NOT NULL,
    Book_ID int NOT NULL,
    Fine_Reason varchar(30),
    Fine_IssueDate date,
    Fine_Status char(10),


    CONSTRAINT fk_fine_borrowbook FOREIGN KEY(Borrow_ID, Book_ID)
REFERENCES BorrowBook(Borrow_ID, Book_ID)
);

CREATE TABLE Donation (
    Donation_ID int PRIMARY KEY,
    Donation_Date date,
    Member_ID int NOT NULL,
    Book_ID int NOT NULL,
    LBR_ID int NOT NULL,
    CONSTRAINT fk_donation_member FOREIGN KEY(Member_ID) REFERENCES
Member(Member_ID),
    CONSTRAINT fk_donation_book FOREIGN KEY(Book_ID) REFERENCES
Book(Book_ID),
    CONSTRAINT fk_donation_librarian FOREIGN KEY(LBR_ID) REFERENCES
Librarian(LBR_ID)
);
```

-AT LEAST,30 MEANİNGFUL SQL SELECT COMMANDS;

```sql
-List all books with their author names and categories.
SELECT
    B.Title AS Book_Title,
    A.Author_FN + ' ' + A.Author_LN AS Author_Name,
    C.Category_Name AS Category,
    B.Publisher
FROM Book B
INNER JOIN Author A ON B.Author_ID = A.Author_ID
INNER JOIN Category C ON B.Category_ID = C.Category_ID;

- List books currently "OnLoan" and their due dates.
SELECT
    M.Member_FN + ' ' + M.Member_LN AS Member_Name,
    B.Title AS Book_Title,
    BB.DueDate,
    DATEDIFF(day, GETDATE(), BB.DueDate) AS Days_Remaining
FROM BorrowBook BB
INNER JOIN Borrow BR ON BB.Borrow_ID = BR.Borrow_ID
INNER JOIN Member M ON BR.Member_ID = M.Member_ID
INNER JOIN Book B ON BB.Book_ID = B.Book_ID
WHERE BB.ItemStatus = 'OnLoan';

- Find the shelf locations of all "Sci-Fi" books.
SELECT
    B.Title,
    B.ISBN,
    SL.Shelf_SectName AS Section,
    SL.Shelf_RowNum AS Row_Number
FROM Book B
INNER JOIN Category C ON B.Category_ID = C.Category_ID
INNER JOIN ShelfLocation SL ON B.Shelf_ID = SL.Shelf_ID
WHERE C.Category_Name = 'Science Fiction'

- Get contact details of members who have "Late" books.
SELECT
    M.Member_FN + ' ' + M.Member_LN AS Full_Name,
    M.Member_PHONE AS Phone,
    M.Member_MAIL AS Email,
    B.Title AS Overdue_Book,
    BB.ReturnDate
FROM BorrowBook BB
JOIN Borrow BR ON BB.Borrow_ID = BR.Borrow_ID
JOIN Member M ON BR.Member_ID = M.Member_ID
JOIN Book B ON BB.Book_ID = B.Book_ID
WHERE BB.ItemStatus = 'Late';

- How many transactions has each librarian managed?
SELECT
    L.LBR_FN + ' ' + L.LBR_LN AS Librarian_Name,
    COUNT(BR.Borrow_ID) AS Total_Transactions
FROM Librarian L
LEFT JOIN Borrow BR ON L.LBR_ID = BR.LBR_ID
GROUP BY L.LBR_FN, L.LBR_LN
ORDER BY Total_Transactions DESC;
```

```sql
- List of donated books and the donor members.
SELECT
    D.Donation_Date,
    M.Member_FN + ' ' + M.Member_LN AS Donor_Name,
    B.Title AS Donated_Book
FROM Donation D
JOIN Member M ON D.Member_ID = M.Member_ID
JOIN Book B ON D.Book_ID = B.Book_ID;

- List books written by authors from the 'UK'.
SELECT
    B.Title,
    A.Author_FN + ' ' + A.Author_LN AS Author,
    B.PublishYear
FROM Book B
JOIN Author A ON B.Author_ID = A.Author_ID
WHERE A.Author_Nation = 'UK'
ORDER BY B.PublishYear DESC;

- Count of books in each category.
SELECT
    C.Category_Name,
    COUNT(B.Book_ID) AS Book_Count
FROM Category C
LEFT JOIN Book B ON C.Category_ID = B.Category_ID
GROUP BY C.Category_Name
ORDER BY Book_Count DESC;

-Top 5 members who borrowed the most books.
SELECT TOP 5
    M.Member_FN + ' ' + M.Member_LN AS Member_Name,
    COUNT(BB.Book_ID) AS Total_Books_Read
FROM Member M
JOIN Borrow BR ON M.Member_ID = BR.Member_ID
JOIN BorrowBook BB ON BR.Borrow_ID = BB.Borrow_ID
GROUP BY M.Member_FN, M.Member_LN
ORDER BY Total_Books_Read DESC;

- Analyze inventory age: Years with more than 5 books published.
SELECT
    PublishYear,
    COUNT(*) AS Total_Books
FROM Book
GROUP BY PublishYear
HAVING COUNT(*) > 5
ORDER BY PublishYear DESC;

- Calculate estimated total debt for "Unpaid" fines (Assuming $5 per day).
SELECT
    SUM(DATEDIFF(DAY, BB.DueDate, BB.ReturnDate) * 5) AS
Estimated_Debt_USD
FROM Fine F
JOIN BorrowBook BB ON F.Borrow_ID = BB.Borrow_ID AND F.Book_ID =
BB.Book_ID
WHERE F.Fine_Status = 'Unpaid';
```

```
- Most popular authors based on borrow count.
SELECT
    A.Author_FN + ' ' + A.Author_LN AS Author,
    COUNT(BB.Borrow_ID) AS Times_Borrowed
FROM Author A
JOIN Book B ON A.Author_ID = B.Author_ID
JOIN BorrowBook BB ON B.Book_ID = BB.Book_ID
GROUP BY A.Author_FN, A.Author_LN
ORDER BY Times_Borrowed DESC;
```

```
- Distribution of books by Publisher.
SELECT
    Publisher,
    COUNT(*) AS Book_Count
FROM Book
GROUP BY Publisher
ORDER BY Book_Count DESC;

- Find members who have NEVER borrowed a book.
SELECT Member_ID, Member_FN, Member_LN
FROM Member
WHERE Member_ID NOT IN (SELECT DISTINCT Member_ID FROM Borrow);

- List 'Available' books written by 'George Orwell'.
SELECT Title, BookStatus
FROM Book
WHERE Author_ID = (SELECT Author_ID FROM Author WHERE Author_FN = 'George'
AND Author_LN = 'Orwell')
AND BookStatus = 'Available';

- Find members who have at least one record in the Fine table.
SELECT DISTINCT M.Member_FN, M.Member_LN
FROM Member M
WHERE EXISTS (
    SELECT 1
    FROM Borrow BR
    JOIN Fine F ON BR.Borrow_ID = F.Borrow_ID
    WHERE BR.Member_ID = M.Member_ID
);
```

- This query lists the first and last names of members along with the total count of books they have read, sorted in descending order.

```
SELECT
    M.Member_FN,
    M.Member_LN,
    COUNT(BB.Book_ID) AS Total_Books_Read
FROM Member M
JOIN Borrow BR ON M.Member_ID = BR.Member_ID
JOIN BorrowBook BB ON BR.Borrow_ID = BB.Borrow_ID
GROUP BY M.Member_FN, M.Member_LN
ORDER BY Total_Books_Read DESC;
```

```sql
-Authors who have written books in 'Sci-Fi' or 'Fantasy'.
SELECT DISTINCT A.Author_FN, A.Author_LN
FROM Author A
JOIN Book B ON A.Author_ID = B.Author_ID
WHERE B.Category_ID IN (SELECT Category_ID FROM Category WHERE
Category_Name IN ('Science Fiction', 'Fantasy'));

- Books that have never been borrowed (Shelf warmers).
SELECT B.Title, B.ISBN
FROM Book B
LEFT JOIN BorrowBook BB ON B.Book_ID = BB.Book_ID
WHERE BB.Borrow_ID IS NULL;


- A unified list of all people (Members and Librarians).
SELECT Member_FN AS First_Name, Member_LN AS Last_Name, 'Member' AS Type
FROM Member
UNION
SELECT LBR_FN AS First_Name, LBR_LN AS Last_Name, 'Staff' AS Type FROM
Librarian
ORDER BY Last_Name;

- Members who are both Borrowers AND Donors.
SELECT Member_ID FROM Borrow
INTERSECT
SELECT Member_ID FROM Donation;

- Members who borrowed books but NEVER donated.
SELECT Member_ID FROM Borrow
EXCEPT
SELECT Member_ID FROM Donation;

- Summary of Available vs OnLoan books.
SELECT 'Available' AS Status, COUNT(*) AS Count FROM Book WHERE BookStatus
= 'Available'
UNION ALL
SELECT 'On Loan' AS Status, COUNT(*) AS Count FROM Book WHERE BookStatus =
'OnLoan';


- Detailed report of fines (Member, Reason, Book, Category).
SELECT
    M.Member_FN + ' ' + M.Member_LN AS Member,
    F.Fine_Reason,
    B.Title AS Book,
    C.Category_Name
FROM Fine F
JOIN Borrow BR ON F.Borrow_ID = BR.Borrow_ID
JOIN Member M ON BR.Member_ID = M.Member_ID
JOIN Book B ON F.Book_ID = B.Book_ID
JOIN Category C ON B.Category_ID = C.Category_ID;
```

```
- Activity log for the last 30 days.
SELECT
    BR.Borrow_Date,
    M.Member_FN AS Member,
    B.Title AS Book,
    L.LBR_FN AS Processed_By
FROM Borrow BR
JOIN BorrowBook BB ON BR.Borrow_ID = BB.Borrow_ID
JOIN Member M ON BR.Member_ID = M.Member_ID
JOIN Book B ON BB.Book_ID = B.Book_ID
JOIN Librarian L ON BR.LBR_ID = L.LBR_ID
WHERE BR.Borrow_Date >= DATEADD(day, -30, GETDATE());


- Total books located in each shelf section.
SELECT
    SL.Shelf_SectName,
    COUNT(B.Book_ID) AS Total_Books
FROM ShelfLocation SL
JOIN Book B ON SL.Shelf_ID = B.Shelf_ID
GROUP BY SL.Shelf_SectName
HAVING COUNT(B.Book_ID) > 0;

- Calculate the percentage of Return Statuses (Returned vs Late).
SELECT
    ItemStatus,
    COUNT(*) AS Count,
    (COUNT(*) * 100.0 / (SELECT COUNT(*) FROM BorrowBook)) AS Percentage
FROM BorrowBook
GROUP BY ItemStatus;

- Find the least read book of the most popular author.
SELECT TOP 1
    B.Title, COUNT(BB.Borrow_ID) as Read_Count
FROM Book B
LEFT JOIN BorrowBook BB ON B.Book_ID = BB.Book_ID
WHERE B.Author_ID = (
    SELECT TOP 1 B2.Author_ID
    FROM BorrowBook BB2
    JOIN Book B2 ON BB2.Book_ID = B2.Book_ID
    GROUP BY B2.Author_ID
    ORDER BY COUNT(*) DESC
)
GROUP BY B.Title
ORDER BY Read_Count ASC;

-Extract email domains and count users per domain.
SELECT
    SUBSTRING(Member_MAIL, CHARINDEX('@', Member_MAIL) + 1,
LEN(Member_MAIL)) AS Domain,
    COUNT(*) AS User_Count
FROM Member
GROUP BY SUBSTRING(Member_MAIL, CHARINDEX('@', Member_MAIL) + 1,
LEN(Member_MAIL));
```

```sql
- Compare the Oldest and Newest book in the library.
SELECT Title, PublishYear, 'Oldest' as Label FROM Book
WHERE PublishYear = (SELECT MIN(PublishYear) FROM Book)
UNION
SELECT Title, PublishYear, 'Newest' as Label FROM Book
WHERE PublishYear = (SELECT MAX(PublishYear) FROM Book);
```

-5 INSER,5 UPDATE,5 DELETE

```sql
- Register a new member. Adds a new user named 'Alice Wonderland' to the system.
INSERT INTO Member (Member_ID, Member_FN, Member_LN, Member_MAIL, Member_ADDR, Member_PHONE)
VALUES (
    5001,
    'Alice',
    'Wonderland',
    'alice.w@lib.co',
    'Rabbit Hole Ave. No:1',
    '5559876'
);

- Add a new book to the inventory. Adds a copy of 'Cosmos' by Carl Sagan.
INSERT INTO Book (Book_ID, Title, Author_ID, Category_ID, ISBN, Publisher, PublishYear, BookCount, Shelf_ID, BookStatus)
VALUES (
    5001,
    'Cosmos',
    1,
    1,
    '978-0345331359',
    'Random House',
    1980,
    5,
    100,
    'Available'
);

- Create a new Borrow Transaction (The Basket). A member comes to the desk to borrow books.
INSERT INTO Borrow (Borrow_ID, Borrow_Date, Member_ID, LBR_ID)
VALUES (
    5001,
    GETDATE(),
    1,
    1
);
```

```
- Hire a new Librarian (Staff Member) Add a new librarian named 'Diana
Prince' to the system who started working today.
INSERT INTO Librarian (LBR_ID, LBR_FN, LBR_LN, LBR_MAIL, LBR_PHONE,
LBR_H_DATE)
VALUES (
    50,
    'Diana',
    'Prince',
    'd.prince@lib.co',
    '5551122',
    GETDATE()
);


- Record a new Donation. Member ID 5 donates a book.
INSERT INTO Donation (Donation_ID, Donation_Date, Member_ID, Book_ID,
LBR_ID)
VALUES (
    1001,
    GETDATE(),
    5,
    5001,
    2
);


- Process a Book Return. Member returned the book. Update status to
'Returned' and set today's date.
UPDATE BorrowBook
SET ReturnDate = GETDATE(),
    ItemStatus = 'Returned'
WHERE Borrow_ID = 1 AND Book_ID = 104;

- Update a Member's Address. Member ID 10 moved to a new house.
UPDATE Member
SET Member_ADDR = 'New York, 5th Avenue Apt 4'
WHERE Member_ID = 10;

- Mark a Book as 'Lost'. A specific book copy was reported lost.
UPDATE Book
SET BookStatus = 'Lost',
    BookCount = BookCount - 1
WHERE Book_ID = 55;

- Pay a Fine. Member paid their fine. Update status from 'Unpaid' to
'Paid'.
UPDATE Fine
SET Fine_Status = 'Paid'
WHERE Fine_ID = 3;

- Bulk Update: Move Books to a New Shelf. All 'Sci-Fi' books (Category 1)
are being moved to Shelf 105.
UPDATE Book
SET Shelf_ID = 105
WHERE Category_ID = 1;
```

```
- Cancel a specific Fine. Deleting a fine entry that was created by
mistake.
DELETE FROM Fine
WHERE Fine_ID = 50;

- Remove a Donation record. Deleting a donation record (perhaps entered in
error).
DELETE FROM Donation
WHERE Donation_ID = 10;

- Remove a specific Book from the catalog. Deleting a book that is damaged
and thrown away. (Note: This only works if no one has borrowed it yet).
DELETE FROM Book
WHERE Book_ID = 9999;

- Delete inactive Members. Deleting a specific member (ID 500) who
requested account closure.
DELETE FROM Member
WHERE Member_ID = 500;

- Remove a specific Transaction Detail. Removing a book from a borrow
basket because the member changed their mind at the last second.
DELETE FROM BorrowBook
WHERE Borrow_ID = 5001 AND Book_ID = 200;
```

-5 ALTER,5 DROP

```
- Add a new column. Add a 'BirthDate' column to the Member table to track
age.
ALTER TABLE Member
ADD Member_BirthDate DATE;

-Modify a column's data type. Increase the size of the Book Title column
from 150 to 250 characters.
ALTER TABLE Book
ALTER COLUMN Title VARCHAR(250);

- Add a Default Constraint. Ensure that if no status is provided for a new
Book, it defaults to 'Available'.
ALTER TABLE Book
ADD CONSTRAINT DF_BookStatus DEFAULT 'Available' FOR BookStatus;

- Drop a column. Remove the 'Publisher' column from the Book table
(Scenario: We no longer track publishers).
ALTER TABLE Book
DROP COLUMN Publisher;

- Add a Check Constraint. Ensure that the 'BookCount' (Stock) never goes
below 0.
ALTER TABLE Book
ADD CONSTRAINT CK_BookCount_Positive CHECK (BookCount >= 0);

- Drop the Fine table. (No other table depends on this).
DROP TABLE Fine;
```

```
-Drop the Donation table. (No other table depends on this).
DROP TABLE Donation;

- Drop the BorrowBook table. (This is a child of Borrow and Book. It must be deleted before them).
DROP TABLE BorrowBook;

- Drop the Borrow table. (Once BorrowBook is gone, we can delete the Borrow header).
DROP TABLE Borrow;

- Drop a temporary or backup table. (Scenario: You created a backup table named 'Member_Backup' and want to delete it).
- Creating a dummy table just to drop it for the example
CREATE TABLE Member_Backup (ID INT, Name VARCHAR(50));

- Now Dropping it
DROP TABLE Member_Backup;
```

At least 10 meaningful indices (except for the primary key) must be created and the corresponding SQL clauses must be shown.

```
-1. Index on Book Title
Reason: The most common operation in a library is searching for a book by its name. This index speeds up searches like WHERE Title LIKE '%Harry Potter%'
CREATE INDEX IX_Book_Title
ON Book (Title);

-2. Index on Member Email (Unique)
Reason: Emails are often used for logging in or contacting members. Making it a UNIQUE index also prevents duplicate member registrations.
CREATE UNIQUE INDEX IX_Member_Email
ON Member (Member_MAIL);

-3. Index on BorrowBook Item Status
Reason: We frequently ran queries to find 'Late' or 'OnLoan' books. This index drastically improves performance for queries like WHERE ItemStatus = 'Late'.
CREATE INDEX IX_BorrowBook_Status
ON BorrowBook (ItemStatus);

-4. Index on Book Category (Foreign Key)
Reason: Used heavily in JOINs to report how many books are in each category (e.g., "Sci-Fi"). Indexing FK columns is best practice.
CREATE INDEX IX_Book_CategoryID
ON Book (Category_ID);

-5. Index on Book Author (Foreign Key)
Reason: Necessary for joining the Book and Author tables efficiently, especially when listing books by a specific author.
CREATE INDEX IX_Book_AuthorID
ON Book (Author_ID);
```

-6. Composite Index on Member Name
Reason: Librarians often search for members by Last Name and then First Name. A composite index covers both ORDER BY and WHERE clauses for names.

```
CREATE INDEX IX_Member_Name
ON Member (Member_LN, Member_FN);
```

-7. Index on Borrow Date
Reason: Critical for reporting queries, such as "List all transactions in the last 30 days" or "Yearly statistics".

```
CREATE INDEX IX_Borrow_Date
ON Borrow (Borrow_Date);
```

-8. Index on Borrow Member (Foreign Key)
Reason: Essential for quickly retrieving the borrowing history of a specific member without scanning the entire transaction table.

```
CREATE INDEX IX_Borrow_MemberID
ON Borrow (Member_ID);
```

-9. Index on Fine Status
Reason: Used to quickly identify who owes money. It optimizes queries like SELECT * FROM Fine WHERE Fine_Status = 'Unpaid'.

```
CREATE INDEX IX_Fine_Status
ON Fine (Fine_Status);
```

-10. Index on BorrowBook DueDate
Reason: Crucial for the daily cron job or report that checks which books are overdue (WHERE DueDate < GETDATE()).

```
CREATE INDEX IX_BorrowBook_DueDate
ON BorrowBook (DueDate);
```

At least 10 views must be created and the corresponding SQL commands must be displayed with the outputs (ie at least 10 CREATE VIEW clauses, 10 SELECT FROM view clauses and the associated outputs).

```sql
--1. View: Master Book Catalogue (vw_BookDetails)
CREATE VIEW vw_BookDetails AS
SELECT
    B.Book_ID,
    B.Title,
    A.Author_FN + ' ' + A.Author_LN AS Author,
    C.Category_Name AS Category,
    B.BookStatus
FROM Book B
INNER JOIN Author A ON B.Author_ID = A.Author_ID
INNER JOIN Category C ON B.Category_ID = C.Category_ID;
SELECT TOP 5 * FROM vw_BookDetails;
```

100 % 1 0

Sonuçlar İletiler

	Book_ID	Title	Author	Category	BookStatus
1	27	War and Peace	Franz Kafka	Science Fiction	Available
2	46	Harry Potter 1	Agatha Christie	Science Fiction	Available
3	63	The Odyssey	J.R.R. Tolkien	Science Fiction	Available
4	103	Hamlet	Dan Brown	Science Fiction	Available
5	104	The Great Gatsby	Jane Austen	Science Fiction	Available

```sql
--2. View: Active Loans (vw_ActiveLoans)
CREATE VIEW vw_ActiveLoans AS
SELECT
    M.Member_FN + ' ' + M.Member_LN AS Member,
    M.Member_MAIL AS Email,
    B.Title AS Book,
    BB.DueDate,
    BB.ItemStatus
FROM BorrowBook BB
JOIN Borrow BR ON BB.Borrow_ID = BR.Borrow_ID
JOIN Member M ON BR.Member_ID = M.Member_ID
JOIN Book B ON BB.Book_ID = B.Book_ID
WHERE BB.ItemStatus IN ('OnLoan', 'Late');
SELECT TOP 5 * FROM vw_ActiveLoans;
```

100 % 1 0

Sonuçlar İletiler

	Member	Email	Book	DueDate	ItemStatus
1	Isabella Martinez	i.martinez421@lib.co	The Da Vinci Code	2025-09-19	OnLoan
2	Liam Williams	l.williams407@lib.co	The Da Vinci Code	2025-10-30	Late
3	Evelyn Moore	e.moore649@lib.co	Madame Bovary	2026-01-18	OnLoan
4	Emma Brown	e.brown329@lib.co	Crime and Punishment	2025-06-14	OnLoan
5	Lucas Hernandez	l.hernandez812@lib.co	Lolita	2025-03-11	Late

```sql
--3. View: Overdue Report (vw_OverdueReport)
CREATE VIEW vw_OverdueReport AS
SELECT
    M.Member_ID,
    M.Member_FN + ' ' + M.Member_LN AS FullName,
    M.Member_PHONE AS Phone,
    B.Title AS OverdueBook,
    DATEDIFF(day, BB.DueDate, GETDATE()) AS Days_Late
FROM BorrowBook BB
JOIN Borrow BR ON BB.Borrow_ID = BR.Borrow_ID
JOIN Member M ON BR.Member_ID = M.Member_ID
JOIN Book B ON BB.Book_ID = B.Book_ID
WHERE BB.ItemStatus = 'Late';
SELECT TOP 5 * FROM vw_OverdueReport;
```

100 % 1 0

Sonuçlar İletiler

	Member_ID	FullName	Phone	OverdueBook	Days_Late
1	407	Liam Williams	5550000407	The Da Vinci Code	78
2	812	Lucas Hernandez	5550000812	Lolita	311
3	538	Olivia Johnson	5550000538	Angels and Demons	155
4	308	Olivia Johnson	5550000308	Brave New World	57
5	302	Noah Jones	5550000302	Angels and Demons	-4

```sql
--4. View: Member Debt Summary (vw_MemberDebts)
CREATE VIEW vw_MemberDebts AS
SELECT
    M.Member_FN + ' ' + M.Member_LN AS Member,
    COUNT(F.Fine_ID) AS Total_Fines,
    'Unpaid' AS Status
FROM Fine F
JOIN Borrow BR ON F.Borrow_ID = BR.Borrow_ID
JOIN Member M ON BR.Member_ID = M.Member_ID
WHERE F.Fine_Status = 'Unpaid'
GROUP BY M.Member_FN, M.Member_LN;
SELECT TOP 5 * FROM vw_MemberDebts ORDER BY Total_Fines DESC;
```

100 % 1 0

Sonuçlar İletiler

	Member	Total_Fines	Status
1	Natasha Romanoff	18	Unpaid
2	Bruce Wayne	18	Unpaid
3	James Smith	17	Unpaid
4	Clark Kent	16	Unpaid
5	Wanda Maximoff	15	Unpaid

```sql
--5. View: Category Statistics (vw_CategoryStats)
CREATE VIEW vw_CategoryStats AS
SELECT
    C.Category_Name,
    COUNT(B.Book_ID) AS Total_Books,
    SUM(CASE WHEN B.BookStatus = 'Available' THEN 1 ELSE 0 END) AS Available_Books
FROM Category C
LEFT JOIN Book B ON C.Category_ID = B.Category_ID
GROUP BY C.Category_Name;
SELECT * FROM vw_CategoryStats;
```

100 % 1 0

Sonuçlar İletiler

	Category_Name	Total_Books	Available_Books
1	Biography	85	85
2	Classic	106	106
3	Fantasy	87	87
4	Horror	108	108
5	Mystery	102	102
6	Philosophy	97	97
7	Psychology	119	119
8	Romance	83	83
9	Science Fiction	105	105
10	Thriller	81	81

```sql
--6. View: Staff Performance (vw_LibrarianActivity)
CREATE VIEW vw_LibrarianActivity AS
SELECT
    L.LBR_FN + ' ' + L.LBR_LN AS Librarian,
    L.LBR_MAIL AS Email,
    COUNT(BR.Borrow_ID) AS Transactions_Handled
FROM Librarian L
LEFT JOIN Borrow BR ON L.LBR_ID = BR.LBR_ID
GROUP BY L.LBR_FN, L.LBR_LN, L.LBR_MAIL;
SELECT * FROM vw_LibrarianActivity;
```

100 % 1 0

Sonuçlar İletiler

	Librarian	Email	Transactions_Handled
1	Diana Prince	d.prince@lib.co	0
2	Ismail Erden	m.freeman@lib.co	763
3	Nusret Ortaç	t.hanks@lib.co	781
4	Taibenur Yavuz	e.watson@lib.co	744
5	Zeliha Mutlu	j.roberts@lib.co	714

```sql
--7. View: Available Inventory (vw_AvailableBooks)
CREATE VIEW vw_AvailableBooks AS
SELECT
    B.Title,
    B.ISBN,
    SL.Shelf_SectName AS Section,
    SL.Shelf_RowNum AS Row
FROM Book B
JOIN ShelfLocation SL ON B.Shelf_ID = SL.Shelf_ID
WHERE B.BookStatus = 'Available';
SELECT TOP 5 * FROM vw_AvailableBooks;
```

100 % 2 0

Sonuçlar İletiler

	Title	ISBN	Section	Row
1	Wuthering Heights	978-100002	Fiction-A	R-01
2	The Catcher in the Rye	978-100005	Fiction-A	R-01
3	It	978-100009	History-B	R-02
4	War and Peace	978-100012	Fiction-A	R-01
5	War and Peace	978-100013	Science-C	R-01

```sql
--8. View: Donation Log (vw_DonationLog)
CREATE VIEW vw_DonationLog AS
SELECT
    D.Donation_Date,
    M.Member_FN + ' ' + M.Member_LN AS Donor,
    B.Title AS Donated_Book,
    L.LBR_LN AS Received_By
FROM Donation D
JOIN Member M ON D.Member_ID = M.Member_ID
JOIN Book B ON D.Book_ID = B.Book_ID
JOIN Librarian L ON D.LBR_ID = L.LBR_ID;
SELECT TOP 5 * FROM vw_DonationLog ORDER BY Donation_Date DESC;
```

100 % 1 0

Sonuçlar İletiler

	Donation_Date	Donor	Donated_Book	Received_By
1	2026-01-16	Harper Thomas	Animal Farm	Erden
2	2026-01-16	Peter Parker	Jane Eyre	Ortaç
3	2026-01-16	Emma Brown	Alice in Wonderland	Erden
4	2026-01-16	Harry Potter	Snow Crash	Yavuz
5	2026-01-16	James Smith	Angels and Demons	Mutlu

```sql
--9. View: Author Directory (vw_AuthorDirectory)
CREATE VIEW vw_AuthorDirectory AS
SELECT
    A.Author_FN + ' ' + A.Author_LN AS Author,
    A.Author_Nation AS Nationality,
    COUNT(B.Book_ID) AS Book_Count
FROM Author A
LEFT JOIN Book B ON A.Author_ID = B.Author_ID
GROUP BY A.Author_FN, A.Author_LN, A.Author_Nation;
SELECT * FROM vw_AuthorDirectory ORDER BY Book_Count DESC;
```

100 % 1 0

Sonuçlar İletiler

	Author	Nationality	Book_Count
1	J.K. Rowling	UK	112
2	Jane Austen	UK	107
3	Victor Hugo	FRA	102
4	Stephen King	USA	99
5	Dan Brown	USA	99
6	George Orwell	UK	98
7	Agatha Christie	UK	96
8	Franz Kafka	AUT	94
9	J.R.R. Tolkien	UK	88
10	F. Dostoevsky	RUS	78

```sql
--10.View: Member Book Counts (vw_MemberCounts)
CREATE VIEW vw_MemberCounts AS
SELECT
    M.Member_ID,
    M.Member_FN,
    COUNT(BB.Borrow_ID) AS Books_Read
FROM Member M
LEFT JOIN Borrow BR ON M.Member_ID = BR.Member_ID
LEFT JOIN BorrowBook BB ON BR.Borrow_ID = BB.Borrow_ID
GROUP BY M.Member_ID, M.Member_FN;
SELECT*FROM vw_MemberCounts
```

100 % 1 0 Sat: 12, Krkt: 1 BŞL CRLF Windows 1254

Sonuçlar İletiler

	Member_ID	Member_FN	Books_Read
1	593	Evelyn	2
2	261	Liam	3
3	925	Ava	0
4	355	Harry	0
5	902	Harper	1
6	238	Isabella	0
7	687	Aiden	0
8	23	Emma	0
9	570	Isabella	1
10	215	James	3
11	879	Bruce	1
12	46	Elijah	0
13	378	Tony	2
14	710	Oliver	1
15	547	Bruce	4
16	401	Ava	1

Sorgu başarıyla yürütüldü. localhost (17.0 RTM) DESKTOP-4763IHM\User (53) LİBRARY1 00:00:00 Satır: 1, Sütun: 1 1.001 satır