College Library Management System

ECS740P – Database Systems

Group 12

Assumptions:

Our assumptions are derived from the case study provided to us. As this is the only system, we have available to us. The goals of the system would be to:

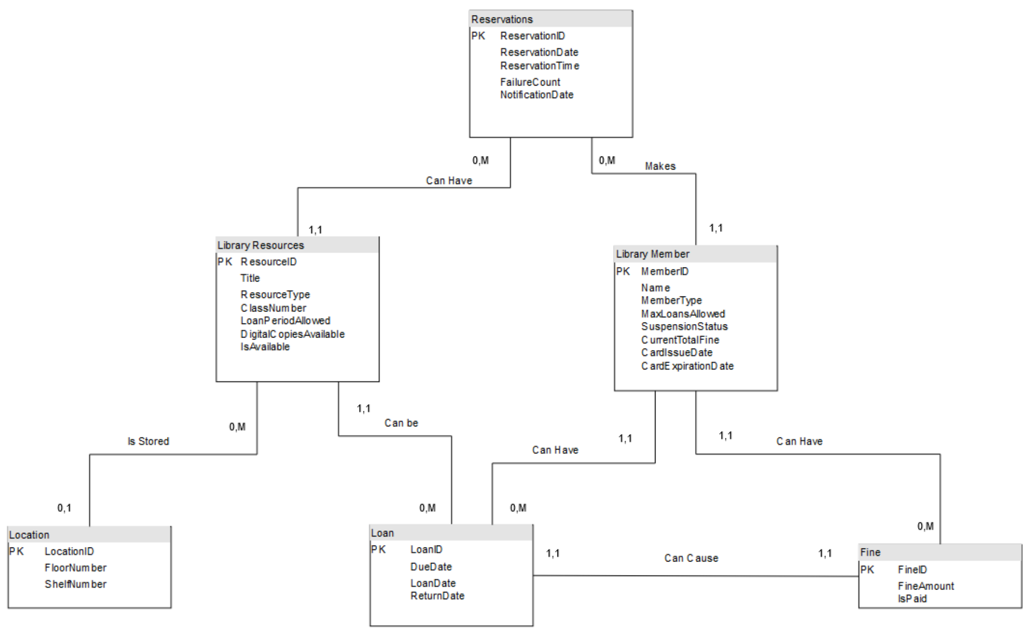
1. Provide a system to loan and reserve books, ebooks and other electronic devices by searching for book titles, resource identification and availability status.
2. Record and manage availability by storing details of loans including maximum loan duration and associated member.
3. Calculate fines and suspensions based on late returns by tracking overdue items on a per loan basis.

This will allow us to build a database for a singular college library system, allowing both members and administrators the ability to access and manage library resources.
Based on this we can begin to derive some assumptions:

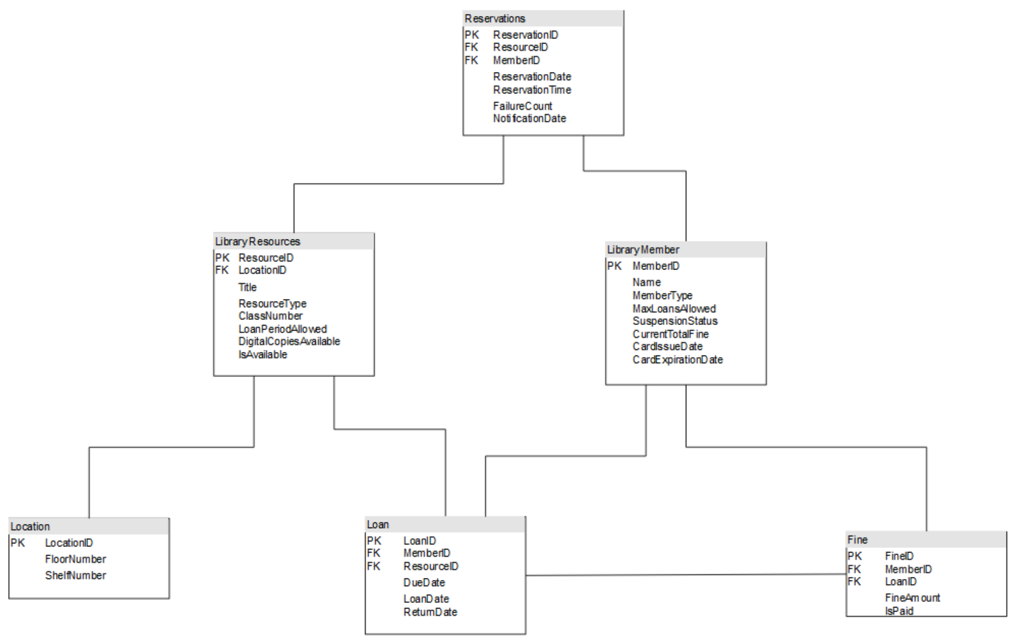
1. Resources can be loaned for three different amounts of time (in days): Zero, Three and Fourteen.
2. Every loan will be taken for the maximum amount of time, determined by the book, but can be returned early.
3. Each resource in the library has a unique resource identifier, eBooks have multiple copies available per resource, physical books of the same title will have a different resource identifier.
4. Library members can loan multiple resources at once, but each resource creates a loan.
5. The cost of each fine is calculated as £1 per day late per loan.
6. Both staff and students can be suspended for exceeding £10 limit.
7. Due date of books will be calculated by adding the allowed loan period to the loan date.
8. Reservations are notified the day before the book due date for previous owner and checked every day until the book is returned, to allow for people to pass.
9. Reservations are booked on a first come first serve basis.
10. Each generally similar subject will share a shelf number. Shelves are not a determined size.
11. Library members have a unique memberID, this is the same as shown on their card.
12. Card expiration date can be 3 or 4 years after the date of issue, depending on what is wanted.
13. No two people make a reservation at the exact same time of the exact same day.
14. There is no cap on number of concurrent reservations made per resource.
15. People can reserve multiple times to take out resource for longer periods.

1a) Conceptual Schema:

- Name is a composite attribute, amalgamating them to show full name in SQL to improve functionality.
- Date also a composite attribute, we will consider it a single value in our diagram for ease.
- Due date, isavailable, Currenttotalfines, suspensionstatus are all derived attributes but they are included in the UML diagram to allow for clarity.



1b) Relational Schema (before normalisation):



1C) NORMALISATION

List all necessary attributes and combine them into a single table to create the universal relation:
| ResourceID | Title | ResourceType | ClassNumber | LoanPeriodAllowed | DigitalCopiesAvailable | IsAvailable | LocationID | FloorNumber | ShelfNumber | MemberID | Name | MemberType | MaxLoansAllowed | SuspensionStatus | CurrentTotalFine | CardIssueDate | CardExpirationDate | ReservationID | ReservationDate | ReservationTime | FailureCount | NotificationDate | LoanID | DueDate | LoanDate | ReturnDate | FineID | FineAmount | Is Paid |

Splitting out the relations by ascertaining their functional dependencies (to give us our groupings):
ResourceID → Title, ResourceType, ClassNumber, LoanPeriodAllowed, DigitalCopiesAvailable, IsAvailable, LocationID
LocationID → FloorNumber, ShelfNumber
MemberID → Name, MemberType, MaxLoansAllowed, SuspensionStatus, CurrentTotalFine, CardIssueDate, CardExpirationDate
ReservationID → ReservationDate, ReservationTime, FailureCount, NotificationDate, ResourceID, MemberID
LoanID → ResourceID, MemberID, DueDate, LoanDate, ReturnDate,
FineID → MemberID, LoanID, FineAmount

List the group entities and identify their most suitable primary keys based on the available candidate keys:
Resources (ResourceID, Title, ResourceType, ClassNumber, LoanPeriodAllowed, DigitalCopiesAvailable, IsAvailable, *LocationID*)
Location (LocationID, FloorNumber, ShelfNumber)
LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, SuspensionStatus, CurrentTotalFine, CardIssueDate, CardExpirationDate)
Reservations (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate, *ResourceID*, *MemberID*)
Loans (LoanID, *ResourceID*, *MemberID*, DueDate, LoanDate, ReturnDate, *FineID*)
Fines (FineID, *MemberID*, *LoanID*, FineAmount, IsPaid)

Define the first normal form:
The first form normalisation is defined as "*A relation in which the intersection of each row and column contains one and only one value.*"(Connelly & Begg, 2015: 573),

Meaning…

- Each column contains atomic values (no multivalued or composite attributes)
- Each entity is uniquely identified by a primary key on each table

1st normal form:

1. Resources(ResourceID, Title, ResourceType, ClassNumber, LoanPeriodAllowed, DigitalCopiesAvailable, IsAvailable, LocationID)
2. Location(LocationID, FloorNumber, ShelfNumber)
3. LibraryMember(MemberID, Name, MemberType, MaxLoansAllowed, SuspensionStatus, CurrentTotalFine, CardIssueDate, CardExpirationDate)
4. Reservation(ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate, ResourceID, MemberID)
5. Loan(LoanID, ResourceID, MemberID, DueDate, LoanDate, ReturnDate)
6. Fine(FineID, LoanID, MemberID, FineAmount, IsPaid)

Define the second normal form:
The second form normalisation is defined as "*A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key'*"(Connelly & Begg, 2015: 578)
Meaning…

- 2NF ensures the table is in 1NF

- And 2NF ensures all non prime attributes depend on the whole primary key, not just part of it (no partial dependency)

All tables are already in 2NF.

Define the third normal form

The third form normalisation is defined as "*A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on the primary key.*" (Connelly & Begg, 2015: 580)
Meaning…

- 3NF ensures the table is in 2NF
- No transitive dependencies exist (non-key attributes depend only on the primary key, not on other non key attributes)

Issues:

- **CurrentTotalFine** depends transitively on the sum of fines in the Fine table.
- **SuspensionStatus** is derivable from CurrentTotalFine (a member is suspended if CurrentTotalFine > £10).
- **IsAvailable** is derived from loan and reservation data (e.g., DigitalCopiesAvailable and active loans/reservations).
- **DueDate** is derivable from LoanDate and LoanPeriod.

Solution:

- Remove CurrentTotalFine and SuspensionStatus from the LibraryMember table.
    - Dynamically calculate CurrentTotalFine as the sum of unpaid fines
    - Dynamically calculate SuspensionStatus with CurrentTotalFine > 10
- Remove IsAvailable from the LibraryResource table.
    - Dynamically calculate:
        - IsAvailable = (TotalCopies - ActiveLoans) > 0; (for physical resources)
        - IsAvailable = (DigitalCopiesAvailable - ActiveLoans) > 0; (for digital resources)
- Remove DueDate from the Loan table.
    - Dynamically calculate: DueDate = LoanDate + LoanPeriod;

Final normalised design:

LibraryResources: (ResourceID, Title, ResourceType, ClassNumber, LoanPeriodAllowed, DigitalCopiesAvailable, *LocationID*)

Location Table: (LocationID, FloorNumber, ShelfNumber)

LibraryMember Table: (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate, CardExpirationDate)

Reservations Table: (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate, *ResourceID, MemberID*)

Loan Table: (LoanID, *ResourceID*, *MemberID,* LoanDate, ReturnDate)

Fine Table: (FineID, *LoanID, MemberID*, FineAmount, IsPaid)

SQL commands for table creation:

```
-- Table for Locations
CREATE TABLE Location (LocationID NUMBER PRIMARY KEY,FloorNumber NUMBER CHECK (FloorNumber >=
0), ShelfNumber NUMBER CHECK (ShelfNumber >= 0));

-- Table for Library Resources
CREATE TABLE LibraryResource (ResourceID NUMBER PRIMARY KEY,Title VARCHAR2(200),ResourceType
VARCHAR2(50),ClassNumber NUMBER,DigitalCopiesAvailable NUMBER CHECK (DigitalCopiesAvailable >=
0),LocationID NUMBER,LoanPeriod NUMBER CHECK (LoanPeriod >= 0),CONSTRAINT fk_resource_location
FOREIGN KEY (LocationID) REFERENCES Location(LocationID));

-- Table for Library Members
CREATE TABLE LibraryMember (MemberID NUMBER PRIMARY KEY,Name VARCHAR2(100),MemberType
VARCHAR2(50),MaxLoansAllowed NUMBER CHECK (MaxLoansAllowed >= 0), CardIssueDate
DATE,CardExpirationDate DATE, CONSTRAINT chk_card_date CHECK (CardExpirationDate > CardIssueDate));

-- Table for Reservations
CREATE TABLE Reservation (ReservationID NUMBER PRIMARY KEY,ReservationDate DATE,ReservationTime
TIMESTAMP,FailureCount NUMBER CHECK (FailureCount >= 0), NotificationDate DATE,ResourceID
NUMBER,MemberID NUMBER,CONSTRAINT fk_reservation_resource FOREIGN KEY (ResourceID)
REFERENCES LibraryResource(ResourceID),CONSTRAINT fk_reservation_member FOREIGN KEY (MemberID)
REFERENCES LibraryMember(MemberID));

-- Table for Loans
CREATE TABLE Loan (LoanID NUMBER PRIMARY KEY,ResourceID NUMBER,MemberID NUMBER,LoanDate
DATE,ReturnDate DATE,CONSTRAINT fk_loan_resource FOREIGN KEY (ResourceID) REFERENCES
LibraryResource(ResourceID),CONSTRAINT fk_loan_member FOREIGN KEY (MemberID) REFERENCES
LibraryMember(MemberID), CONSTRAINT chk_return_date CHECK (ReturnDate > LoanDate));

-- Table for Fines
CREATE TABLE Fine (FineID NUMBER PRIMARY KEY,LoanID NUMBER,MemberID NUMBER,FineAmount
NUMBER CHECK (FineAmount >= 0), IsPaid CHAR(1) CHECK (IsPaid IN ('Y', 'N')), CONSTRAINT fk_fine_loan
FOREIGN KEY (LoanID) REFERENCES Loan(LoanID),CONSTRAINT fk_fine_member FOREIGN KEY (MemberID)
REFERENCES LibraryMember(MemberID) );
```

Data population in tables:

```
--LibraryMember
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (1, 'Alice Smith', 'Student', 5, TO_DATE('2021-09-01', 'YYYY-MM-DD'), TO_DATE('2024-09-01', 'YYYY-
MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (2, 'John Doe', 'Staff', 10, TO_DATE('2020-05-01', 'YYYY-MM-DD'), TO_DATE('2025-05-01', 'YYYY-MM-
DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (3, 'Mary Johnson', 'Student', 5, TO_DATE('2022-01-01', 'YYYY-MM-DD'), TO_DATE('2025-01-01', 'YYYY-
MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (4, 'Grant Permission', 'Staff', 10, TO_DATE('2021-07-01', 'YYYY-MM-DD'), TO_DATE('2026-07-01',
'YYYY-MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
```

```
VALUES (5, 'Emma Davis', 'Student', 5, TO_DATE('2022-08-01', 'YYYY-MM-DD'), TO_DATE('2025-08-01', 'YYYY-
MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (6, 'Robert Lee', 'Staff', 10, TO_DATE('2020-04-01', 'YYYY-MM-DD'), TO_DATE('2023-04-01', 'YYYY-MM-
DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (7, 'Justin Time', 'Student', 5, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2026-01-01', 'YYYY-
MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (8, 'Phil Ter Data', 'Staff', 10, TO_DATE('2022-02-01', 'YYYY-MM-DD'), TO_DATE('2025-02-01', 'YYYY-
MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (9, 'Norm Alized', 'Student', 5, TO_DATE('2021-03-01', 'YYYY-MM-DD'), TO_DATE('2024-03-01', 'YYYY-
MM-DD'));
INSERT INTO LibraryMember (MemberID, Name, MemberType, MaxLoansAllowed, CardIssueDate,
CardExpirationDate)
VALUES (10, 'William Young', 'Staff', 10, TO_DATE('2020-12-01', 'YYYY-MM-DD'), TO_DATE('2023-12-01', 'YYYY-
MM-DD'));

--Location
INSERT INTO Location (LocationID,FloorNumber,ShelfNumber) VALUES (101,1,1);
INSERT INTO Location (LocationID,FloorNumber,ShelfNumber) VALUES (201,2,1);
INSERT INTO Location (LocationID,FloorNumber,ShelfNumber) VALUES (103,1,3);
INSERT INTO Location (LocationID,FloorNumber,ShelfNumber) VALUES (104,1,4);
INSERT INTO Location (LocationID,FloorNumber,ShelfNumber) VALUES (301,3,1);
INSERT INTO Location (LocationID,FloorNumber,ShelfNumber) VALUES (0,0,0);

--Resources
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1001, 'GROUP by Regret, ORDER BY Poor Decisions', 'Book',10,0,101,14);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1002, 'The Very Hungry Caterpillar ', 'Book',11,0,201,14);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1003, 'Principia Mathematica', 'Book',12,0, 103, 0);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1004, 'The Concept of Law', 'eBook',13, 7, 0, 14);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1005, 'The Feynman Lectures on Physics', 'Book',14,0, 104, 3);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1006, 'The Selfish Gene', 'eBook', 15,10,0, 3);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1007, 'Laptop1', 'Laptop', 0,0,301, 0);
INSERT INTO LibraryResource (ResourceID, Title, ResourceType, ClassNumber, DigitalCopiesAvailable,
LocationID, LoanPeriod)
VALUES (1008, 'The Joy of SQL', 'Book',10,0,101, 0);
```

```
--Loan
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10001, 1001, 3,TO_DATE('2024-08-01', 'YYYY-MM-DD'), TO_DATE('2024-08-16', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10002, 1002,3, TO_DATE('2024-07-15', 'YYYY-MM-DD'), TO_DATE('2024-07-29', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10003, 1002, 5, TO_DATE('2024-08-05', 'YYYY-MM-DD'), TO_DATE('2024-08-19', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10004, 1002, 6, TO_DATE('2024-09-01', 'YYYY-MM-DD'), TO_DATE('2024-09-15', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10005, 1004,1, TO_DATE('2024-06-10', 'YYYY-MM-DD'), TO_DATE('2024-06-23', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10006, 1001, 3, TO_DATE('2024-05-20', 'YYYY-MM-DD'), TO_DATE('2024-06-02', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10007, 1006, 2, TO_DATE('2024-07-01', 'YYYY-MM-DD'), TO_DATE('2024-07-04', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10008, 1004, 2, TO_DATE('2024-08-10', 'YYYY-MM-DD'), TO_DATE('2024-08-24', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10009, 1005, 2, TO_DATE('2024-09-05', 'YYYY-MM-DD'), TO_DATE('2024-09-10', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10010, 1005, 10, TO_DATE('2024-05-25', 'YYYY-MM-DD'), TO_DATE('2024-05-28', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10011,1004,7, TO_DATE('2024-06-15', 'YYYY-MM-DD'), TO_DATE('2024-06-29', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10012, 1004, 7,TO_DATE('2024-07-20', 'YYYY-MM-DD'), TO_DATE('2024-08-03', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10013, 1004, 8,  TO_DATE('2024-08-25', 'YYYY-MM-DD'), TO_DATE('2024-09-08', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10014,1004,3, TO_DATE('2024-11-26', 'YYYY-MM-DD'),  NULL);
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10015, 1002,1, TO_DATE('2024-11-20', 'YYYY-MM-DD'),NULL);
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10016, 1006, 5, TO_DATE('2024-07-02', 'YYYY-MM-DD'), TO_DATE('2024-07-05', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10017, 1005, 9, TO_DATE('2024-10-02', 'YYYY-MM-DD'), TO_DATE('2024-10-30', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10018, 1001, 8, TO_DATE('2024-11-08', 'YYYY-MM-DD'), TO_DATE('2024-11-16', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10019, 1001, 4, TO_DATE('2024-11-16', 'YYYY-MM-DD'), NULL);
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10020, 1005, 4, TO_DATE('2024-10-30', 'YYYY-MM-DD'), TO_DATE('2024-11-02', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10021, 1001, 4, TO_DATE('2024-10-25', 'YYYY-MM-DD'), TO_DATE('2024-11-08', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10022, 1005, 6, TO_DATE('2024-11-02', 'YYYY-MM-DD'), TO_DATE('2024-11-06', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10023, 1005, 2, TO_DATE('2024-11-06', 'YYYY-MM-DD'), TO_DATE('2024-11-10', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10024, 1005, 9, TO_DATE('2024-11-10', 'YYYY-MM-DD'), TO_DATE('2024-11-13', 'YYYY-MM-DD'));
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10025, 1004, 6, TO_DATE('2024-11-27', 'YYYY-MM-DD'),NULL);
INSERT INTO Loan (LoanID, ResourceID, MemberID, LoanDate, ReturnDate)
VALUES (10026, 1005, 2, TO_DATE('2024-11-22', 'YYYY-MM-DD'),NULL);

--Fine
INSERT INTO Fine (FineID,LoanID,MemberID,FineAmount,IsPaid) VALUES (1,10001,3,1.00,'Y');
INSERT INTO Fine (FineID,LoanID,MemberID,FineAmount,IsPaid) VALUES (2,10009,2,2.00,'N');
INSERT INTO Fine (FineID,LoanID,MemberID,FineAmount,IsPaid) VALUES (3,10017,3,25.00,'N');
INSERT INTO Fine (FineID,LoanID,MemberID,FineAmount,IsPaid) VALUES (4,10022,6,1.00,'Y');
INSERT INTO Fine (FineID,LoanID,MemberID,FineAmount,IsPaid) VALUES (5,10023,2,1.00,'N');

--Reservations
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (1, TO_DATE('2024-11-01', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-01 09:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 3, TO_DATE('2024-11-09', 'YYYY-MM-DD'), 1005, 7);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (2, TO_DATE('2024-11-01', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-01 11:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 0, TO_DATE('2024-11-01', 'YYYY-MM-DD'), 1005, 6);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (3, TO_DATE('2024-11-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-03 14:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 1, TO_DATE('2024-11-09', 'YYYY-MM-DD'), 1005, 9);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (4, TO_DATE('2024-11-04', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-04 18:27:01', 'YYYY-MM-DD
HH24:MI:SS'), 0, TO_DATE('2024-11-05', 'YYYY-MM-DD'), 1005, 2);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (5, TO_DATE('2024-11-27', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-27 19:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 0,NULL, 1002, 6);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (6, TO_DATE('2024-11-05', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-05 12:27:16', 'YYYY-MM-DD
HH24:MI:SS'), 0, TO_DATE('2024-11-07', 'YYYY-MM-DD'), 1001, 8);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (7, TO_DATE('2024-11-13', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-13 13:31:21', 'YYYY-MM-DD
HH24:MI:SS'), 0, TO_DATE('2024-11-16', 'YYYY-MM-DD'), 1001, 4);
INSERT INTO Reservation (ReservationID, ReservationDate, ReservationTime, FailureCount, NotificationDate,
ResourceID, MemberID)
VALUES (8, TO_DATE('2024-11-22', 'YYYY-MM-DD'), TO_TIMESTAMP('2024-11-22 09:00:00', 'YYYY-MM-DD
HH24:MI:SS'), 0,NULL, 1001, 3);
```

SQL views:

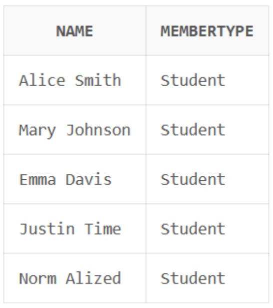
```
/*  This view provides a summary of all current loans, indicating whether they are overdue. */
CREATE OR REPLACE VIEW v_CurrentLoans AS SELECT L.LoanID,L.ResourceID,M.Name AS
MemberName,L.LoanDate,CASE WHEN L.ReturnDate IS NULL AND L.LoanDate+R.LoanPeriod <
SYSDATE THEN 'Yes' ELSE 'No' END AS IsOverdue,L.LoanDate+R.LoanPeriod AS DateToReturn,R.title AS
ResourceTitle FROM Loan L JOIN LibraryMember M ON L.MemberID=M.MemberID JOIN LibraryResource
R ON L.ResourceID=R.ResourceID WHERE L.ReturnDate IS NULL;

/* This view summarises the activity of each library member, detailing their current loans, reservations,
any outstanding fines and suspension status. */
CREATE OR REPLACE VIEW v_MemberActivityOverview AS SELECT
M.MemberID,M.Name,M.MemberType,COUNT(DISTINCT vL.LoanID) AS
TotalCurrentLoans,COUNT(DISTINCT R.ReservationID) AS TotalReservations,SUM(CASE WHEN
F.IsPaid='N' THEN F.FineAmount ELSE 0 END) AS CurrentTotalFine,CASE WHEN SUM(CASE WHEN
F.IsPaid='N' THEN F.FineAmount ELSE 0 END)>10 THEN 'Suspended' ELSE 'Not Suspended' END AS
SuspensionStatus,MAX(L.LoanDate+Rs.LoanPeriod) AS LatestDueDate,MAX(R.ReservationDate) AS
LatestReservationDate FROM LibraryMember M LEFT JOIN Loan L ON M.MemberID=L.MemberID LEFT
JOIN Reservation R ON M.MemberID=R.MemberID LEFT JOIN LibraryResource Rs ON
```

```
L.ResourceID=Rs.ResourceID LEFT JOIN Fine F ON L.LoanID=F.LoanID LEFT JOIN v_CurrentLoans vL ON
L.LoanID=vL.LoanID GROUP BY M.MemberID,M.Name,M.MemberType;

/* This view provides information about all resources, including their availability status and locations. */
CREATE OR REPLACE VIEW v_ResourceAvailability AS SELECT
R.ResourceID,R.title,R.ResourceType,R.DigitalCopiesAvailable,R.LocationID,CASE WHEN
R.ResourceType!='Book' THEN R.DigitalCopiesAvailable-(SELECT COUNT(*) FROM v_CurrentLoans vL
WHERE vL.ResourceID=R.ResourceID) ELSE NULL END AS AvailableCopies,CASE WHEN
R.ResourceType!='Book' AND EXISTS (SELECT 1 FROM v_CurrentLoans vL WHERE
vL.ResourceID=R.ResourceID GROUP BY vL.ResourceID HAVING COUNT(*)>=R.DigitalCopiesAvailable)
THEN 'Not Available' WHEN R.ResourceType='Book' AND EXISTS (SELECT 1 FROM v_CurrentLoans vL
WHERE vL.ResourceID=R.ResourceID) THEN 'Not Available' ELSE 'Available' END AS Availability,(SELECT
vL.DateToReturn FROM v_CurrentLoans vL WHERE vL.ResourceID=R.ResourceID AND vL.DateToReturn IS
NOT NULL FETCH FIRST 1 ROWS ONLY) AS ReturnDate,L.FloorNumber,L.ShelfNumber FROM
LibraryResource R JOIN Location L ON R.LocationID=L.LocationID GROUP BY
R.ResourceID,R.Title,R.ResourceType,R.DigitalCopiesAvailable,R.LocationID,L.FloorNumber,L.ShelfNum
ber;

/*This view shows all outstanding fines owed by members, providing details on what caused the fine.*/
CREATE OR REPLACE VIEW v_FinesDue AS SELECT F.FineID,M.Name AS
MemberName,F.LoanID,F.FineAmount,F.IsPaid,L.LoanDate,L.LoanDate+R.LoanPeriod AS
DueDate,L.ReturnDate FROM Fine F JOIN Loan L ON F.LoanID=L.LoanID JOIN LibraryResource R ON
L.ResourceID=R.ResourceID JOIN LibraryMember M ON L.MemberID=M.MemberID WHERE F.IsPaid='N';
```

SQL queries:

Simple queries:
1) Retrieve all student member names

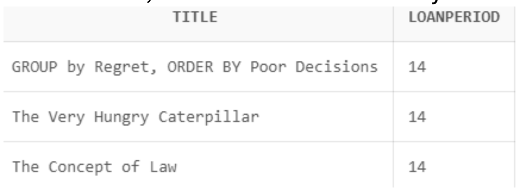
```
SELECT Name,MemberType FROM LibraryMember WHERE MemberType='Student';
```



NAME	MEMBERTYPE
Alice Smith	Student
Mary Johnson	Student
Emma Davis	Student
Justin Time	Student
Norm Alized	Student

2) Find resources with 14-day loan periods:

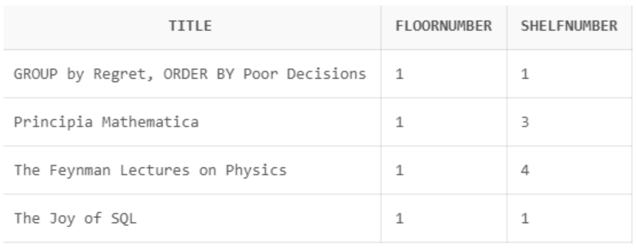
```
SELECT Title,LoanPeriod FROM LibraryResource WHERE LoanPeriod=14;
```



TITLE	LOANPERIOD
GROUP by Regret, ORDER BY Poor Decisions	14
The Very Hungry Caterpillar	14
The Concept of Law	14

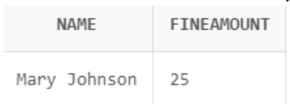
3) Get titles and locations for books on the first floor

SELECT R.Title,L.FloorNumber,L.ShelfNumber FROM LibraryResource R JOIN Location L ON R.LocationID=L.LocationID WHERE L.FloorNumber=1 AND R.ResourceType='Book';



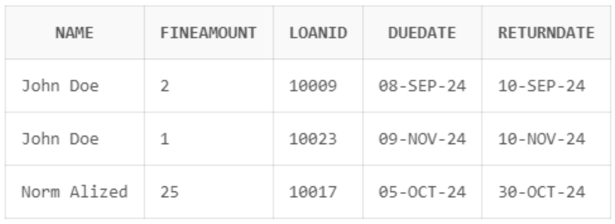
TITLE	FLOORNUMBER	SHELFNUMBER
GROUP by Regret, ORDER BY Poor Decisions	1	1
Principia Mathematica	1	3
The Feynman Lectures on Physics	1	4
The Joy of SQL	1	1

4) Find members with fines over 5 Pounds
SELECT M.Name,F.FineAmount FROM LibraryMember M JOIN Fine F ON M.MemberID=F.MemberID WHERE F.FineAmount>5;



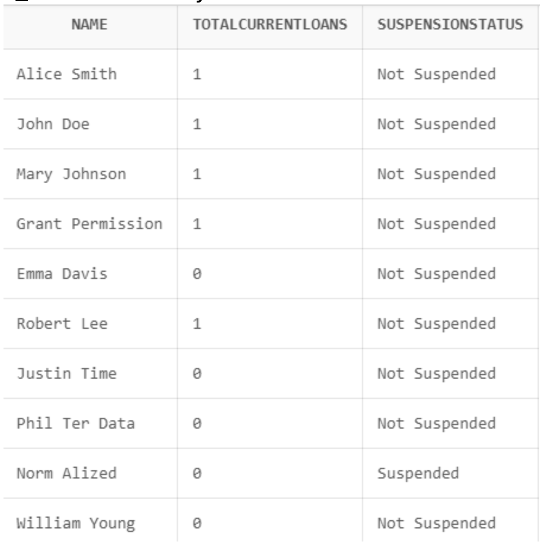
NAME	FINEAMOUNT
Mary Johnson	25

Intermediate queries:
1) List unpaid fine details by member, including member name, fine amount, and loan details
SELECT M.Name,F.FineAmount,F.LoanID,F.DueDate,F.ReturnDate FROM v_FinesDue F JOIN LibraryMember M ON F.MemberName=M.Name;



NAME	FINEAMOUNT	LOANID	DUEDATE	RETURNDATE
John Doe	2	10009	08-SEP-24	10-SEP-24
John Doe	1	10023	09-NOV-24	10-NOV-24
Norm Alized	25	10017	05-OCT-24	30-OCT-24

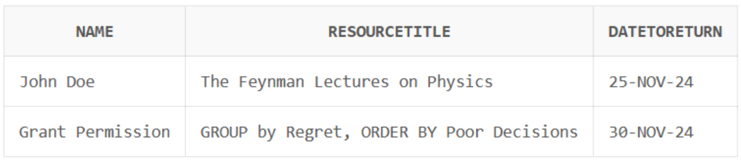
2) Retrieve current loan count and suspension status for all library members
SELECT M.Name,MA.TotalCurrentLoans,MA.SuspensionStatus FROM LibraryMember M JOIN v_MemberActivityOverview MA ON M.MemberID=MA.MemberID;



NAME	TOTALCURRENTLOANS	SUSPENSIONSTATUS
Alice Smith	1	Not Suspended
John Doe	1	Not Suspended
Mary Johnson	1	Not Suspended
Grant Permission	1	Not Suspended
Emma Davis	0	Not Suspended
Robert Lee	1	Not Suspended
Justin Time	0	Not Suspended
Phil Ter Data	0	Not Suspended
Norm Alized	0	Suspended
William Young	0	Not Suspended

3) List overdue loans including member name, resource title, and return date.

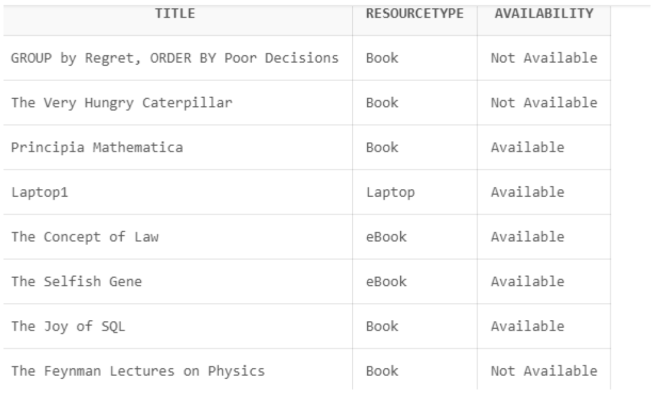
```
SELECT M.Name,CL.ResourceTitle,CL.DateToReturn FROM v_CurrentLoans CL JOIN LibraryMember M
ON CL.MemberName=M.Name WHERE CL.IsOverdue='Yes';
```



NAME	RESOURCETITLE	DATETORETURN
John Doe	The Feynman Lectures on Physics	25-NOV-24
Grant Permission	GROUP by Regret, ORDER BY Poor Decisions	30-NOV-24

4) Check resource availability including title, resource type, and availability status.

```
SELECT R.Title,R.ResourceType,VA.Availability FROM LibraryResource R JOIN v_ResourceAvailability VA
ON R.ResourceID=VA.ResourceID;
```

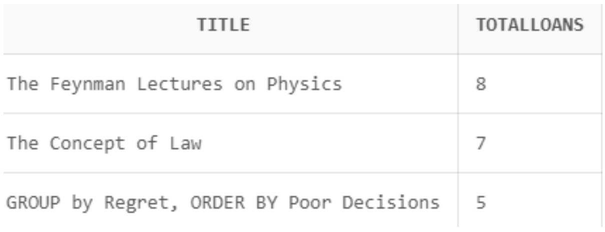


TITLE	RESOURCETYPE	AVAILABILITY
GROUP by Regret, ORDER BY Poor Decisions	Book	Not Available
The Very Hungry Caterpillar	Book	Not Available
Principia Mathematica	Book	Available
Laptop1	Laptop	Available
The Concept of Law	eBook	Available
The Selfish Gene	eBook	Available
The Joy of SQL	Book	Available
The Feynman Lectures on Physics	Book	Not Available

Advanced queries:

1) Retrieve the top 3 most popular library resources based on loan count

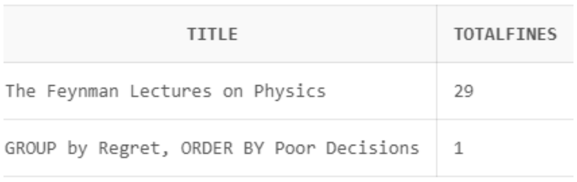
```
SELECT R.Title,COUNT(L.LoanID) AS TotalLoans FROM LibraryResource R JOIN Loan L ON
R.ResourceID=L.ResourceID GROUP BY R.Title ORDER BY TotalLoans DESC FETCH FIRST 3 ROWS ONLY;
```



TITLE	TOTALLOANS
The Feynman Lectures on Physics	8
The Concept of Law	7
GROUP by Regret, ORDER BY Poor Decisions	5

2) Calculate total cost of fines generated for each library resource

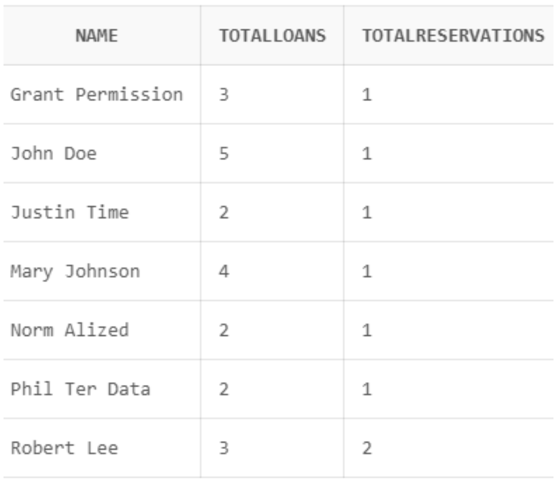
```
SELECT R.Title,SUM(F.FineAmount) AS TotalFines FROM LibraryResource R JOIN Loan L ON
R.ResourceID=L.ResourceID JOIN Fine F ON L.LoanID=F.LoanID GROUP BY R.Title;
```



TITLE	TOTALFINES
The Feynman Lectures on Physics	29
GROUP by Regret, ORDER BY Poor Decisions	1

3) Summarise member loans and reservations for active members (more than one loan and at least one reservation)

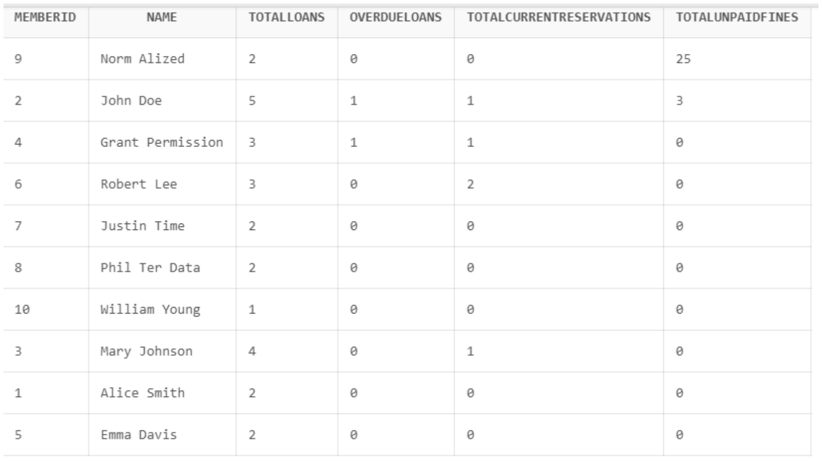
```
SELECT M.Name,COUNT(DISTINCT L.LoanID) AS TotalLoans,COUNT(DISTINCT R.ReservationID) AS
TotalReservations FROM LibraryMember M LEFT JOIN Loan L ON M.MemberID=L.MemberID LEFT JOIN
Reservation R ON M.MemberID=R.MemberID GROUP BY M.Name HAVING COUNT(DISTINCT L.LoanID)>1
AND COUNT(DISTINCT R.ReservationID)>0;
```



NAME	TOTALLOANS	TOTALRESERVATIONS
Grant Permission	3	1
John Doe	5	1
Justin Time	2	1
Mary Johnson	4	1
Norm Alized	2	1
Phil Ter Data	2	1
Robert Lee	3	2

4) Analyse detailed member activity: total loans, overdue loans, active reservations, and unpaid fines

```
SELECT M.MemberID,M.Name,COUNT(DISTINCT L.LoanID) AS TotalLoans,SUM(CASE WHEN
L.ReturnDate IS NULL AND L.LoanDate+R.LoanPeriod<CURRENT_DATE THEN 1 ELSE 0 END) AS
OverdueLoans,COUNT(DISTINCT CASE WHEN L.ReturnDate IS NULL THEN Re.ReservationID END) AS
TotalCurrentReservations,SUM(CASE WHEN F.IsPaid='N' THEN F.FineAmount ELSE 0 END) AS
TotalUnpaidFines FROM LibraryMember M LEFT JOIN Loan L ON M.MemberID=L.MemberID LEFT JOIN
LibraryResource R ON L.ResourceID=R.ResourceID LEFT JOIN Fine F ON L.LoanID=F.LoanID LEFT JOIN
Reservation Re ON M.MemberID=Re.MemberID GROUP BY M.MemberID,M.Name HAVING
COUNT(DISTINCT L.LoanID)>0 OR COUNT(DISTINCT Re.ReservationID)>0 ORDER BY TotalUnpaidFines
DESC,OverdueLoans DESC;
```



MEMBERID	NAME	TOTALLOANS	OVERDUELOANS	TOTALCURRENTRESERVATIONS	TOTALUNPAIDFINES
9	Norm Alized	2	0	0	25
2	John Doe	5	1	1	3
4	Grant Permission	3	1	1	0
6	Robert Lee	3	0	2	0
7	Justin Time	2	0	0	0
8	Phil Ter Data	2	0	0	0
10	William Young	1	0	0	0
3	Mary Johnson	4	0	1	0
1	Alice Smith	2	0	0	0
5	Emma Davis	2	0	0	0

Disclaimers:

- Someone who is currently suspended could take out a book without any problems.
- People could also loan more books than they are allowed.
- People could technically take out a loan for a resource that is already on loan.
- People with expired cards can also take out books.
- Fines aren't automatically generated for overdue loans, neither is the cost of the fine.

<u>Bibliography</u>

Connolly, T., & Begg, C. (2019). *Database systems: A practical approach to design, implementation, and management* (6th ed.). Pearson Education.