Library Database System Design

Table of Contents:

Part 1: Database Design

1.1 List of Assumptions

Resource handling:

1. The library has two categories of resources: devices and books.
    a. Devices have 3 types: e-book reader (i.e., e-reader), tablet, and laptop. They are stored in the Devices table.
    b. Books have 2 formats: physical and e-book. Both types fall under the “Books” table.
        i. Each book has a unique ISBN (stored as a string), preferably in the ISBN-13 format. However, the Books table will use an incrementing integer as the primary key to address two issues: (1) ISBN, a natural key, may change in the future, and (2) flexibility is needed for older ISBN-10 numbers.
        ii. E-books fare like physical books – there is a finite amount of licenses per e-book that can be loaned out at any given time.
    c. The library system organises its resources through two main tables: “Resources” and “Copies” tables. "Resources" table contains metadata about all resources, illustrating how they function within the library. In contrast, the "Copies" table tracks individual instances of each resource. Book to Resources relationship is 1:1; and Books to Devices, 1:1. Resources to Copies is 1:N. Separation of resource metadata from physical copies enhances scalability, minimises data redundancy, & promotes efficient data management practices.
    d. The ResourceID primary key for the Resources table will be an incrementing integer for simplicity.
2. The class number is independent of loan length and is derived from the Dewey Decimal System.
    a. Format: 3 digit – XYY respectively as X (general category), YY subcategory
    b. Class numbers don't directly correlate with college courses, but they can help students find relevant resources in specific fields. Only books have class numbers, devices do not.
3. Each resource has a location. The location format is X-YYY where X is the floor number, YYY is the shelf number.
    a. E-books have the location X-000, which indicates it is online
    b. The library has 3 floors, so physical books and devices will have the locations 0-YYY, 1-YYY or 2-YYY.

Loan mechanics:

4. Each resource has a specific loan period, indicating how long it can be borrowed before being returned.
5. Some resources, like historical texts, are only available in the library and have a loan period of 0. These resources are dealt with in person and as such, any in-house hourly loans aren’t reflected in the database.
6. Devices have a default loan period of 3 days.
7. Books have a default loan period of 21 days, which can be adjusted based on demand at the academic year’s end. Popularity is determined by the number of reservations in the previous year and resets every September. If a book is above a certain popularity threshold (subject to management, and not the database), its loan period is updated to 3 days; otherwise, it remains at 21 days.
8. All users of the library are students and staff of the same college. They are validated with the use of library cards reflected in the Users table. Loan limits: 5 resources at once for students; and for staff, up to 10.

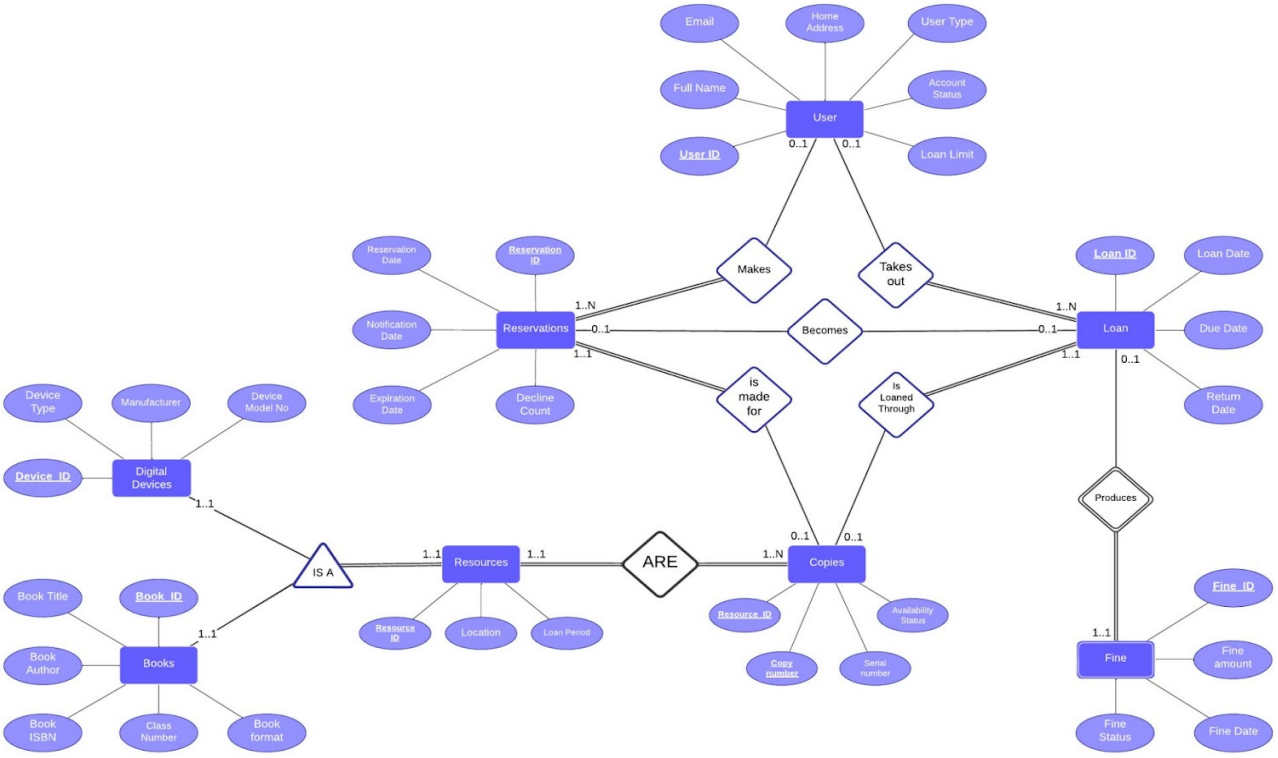
Fine mechanics:

9. If a user fails to return a resource by the due date, a £1 daily fine per overdue resource item is incurred.
10. If a user owes more than £10 in overdue fines for **any** loaned resource, they are suspended. To be reinstated, they must return all items and pay all fines, even if other fines are not over £10. The system tracks suspended users and their outstanding fines.
11. Fines are capped as follows for **delayed returns**:
    a. books (*excluding* e-books): £50
    b. e-readers: £50
    c. tablets: £150
    d. laptops: £250
12. If an item cannot be returned due to loss or damage, the user must pay a fine which is **double the delayed returns fine cap** for the respective resource sub-type. The rest of the resource’s value is covered by the library’s insurance.
13. If a student completes a course but has an outstanding fine, they cannot graduate or advance to the next school year until the fine is paid. Graduated students will be removed from the system to minimise database size, as the complete student database is managed separately. Thus, the “Users” table will only include current staff and students permitted to access library services during the academic year and term breaks.
14. Fines cannot be partially paid, they must be paid in full. All previous fines are kept there.
15. We monitor fines (lost or damaged/delayed) through the Fines table based on the return date. A marked return date indicates a late return. If a resource is damaged or lost, no return date is *reco*rded since it can't be returned to the system. Once repaired, library staff must re-enter the resource and copy number as a new item.

Loaning and reserving:

16. The availability of a resource is determined by the copies and loans table. A resource copy is considered out for loan if there's a copy in the loan table without a return date. Once a return date is added, the resource is marked as returned and available. If a copy isn't *listed* in the loan table, it means it hasn't been borrowed and has been available since entering the library system.
17. When a user attempts to reserve a resource, the system will check whether it is available by running through the system if the resource in question has *any available copies* of it in the library. If it is, a loan is made. If not, a reservation is made.
    a. We distinguish between a loan (where a user has successfully taken out a resource) and a reservation (the user has been put on the waiting list and will be notified when the resource is available).
    b. Reservations operate on a first-come, first-served basis. The user with the earliest reservation will be notified first when a loan becomes available. Notifications last 24 hours; if unacknowledged, this is deemed as declined. The user is removed and readded to the end of the queue with a new reservation ID.
    c. If the user declines the offer of a loan or does change their reservation to a loan in 3 days, the offer is made to the next user in the waiting list, and the initial user's reservation is moved to the end of the waiting list.
    d. The maximum amount of declines is 3 times. Else, the reservation is cancelled and removed from the table.
    e. Users can make unlimited reservations, but if they have the maximum active loans, they must return an item before accepting a new loan.

1.2 Conceptual ER Diagram



The diagram's derivation in Chen notation reflects the library requirements and our assumptions about the system. We began by identifying entities from the system requirements, consequently, establishing the ER diagram structure. We aimed to define tables separately for (1) the metadata, i.e., the concept of certain entities that live in the library system, to reduce duplication, and (2) the physical instances in which we will need to track conceptual entities.

Upon creating our assumptions, our core entities were determined to be: User, Reservations, Copies, Resources, Digital Devices, Books, Loans and Fines. Once these were established, we moved to identifying attributes and relationships.

Attributes were determined by analysing system requirements and linking them to the right entities. Shared attributes like Loan Period and Location were placed in a generalized entity (Resources), while type-specific attributes (e.g., Book Author, Device Manufacturer) remained in their respective tables (Book/Device).

To analyse relationships within our system, we examined how entities interacted with each other, and their overall impact on the system. For instance, we found that the relationship between Resources and Copies was crucial in tracking individual copies of books and devices. Since each instance of a book or device can have many copies, we created a Copies table to store this information,

allowing the Resources table to retain relevant metadata for all copies. While this may seem like a trivial distinction, this relationship fundamentally shaped the structure of our database, enhancing flexibility.

Key points regarding our database:

- There is one weak entity, Fines, since a resource can be returned without incurring a fine
- Resource type is a view (see part 2) not an attribute, to prevent duplication in the Copies, Resources, Book, & Device tables.

1.3 Relational Schema [1]

IMPORTANT NOTE: The relational schema diagram has been removed from this submission on Michael S.' instruction in class (Week 10 lecture). If this diagram is indeed required, please let us know and we will provide it.

Entity	***Attributes***
Users	User_ID (PK), User_Full_Name, User_Email, User_Home_Address, User_Type, User_Account_Status
Loans	Loan_ID (PK), Loan_Date, Loan_Due_Date, Loan_Return_Date, *User_ID* (FK *to User table*), *Copy_Num* (FK *to Copies table*), *Resource_ID* (FK *to Copies table*)
Fines	Fine_ID (PK), *Loan_ID* (FK *to Loans table*), Fine_Amt, Fine_Status, Fine_Date
Resources	Resource_ID (PK), Resource_Loan_Pd, Resource_Location
Copies	Copy_Num (PK), *Resource_ID* (PK & FK *to Resources table*), Copy_Avail_Stat, Device_Serial_Num
Reservations	Reservation_ID (PK), Reserv_Date, Reserv_Notif_Date, Reserv_Expir_Date, Reserv_Decline_Count, *User_ID* (FK to Users table), *Copy_Num* (FK to Copies table), *Resource_ID* (FK to Copies table)
Books	Book_ID (PK), *Resource_ID* (FK *to Copies table*), Book_Format, Book_Author, Book_Title, Book_ISBN, Book_Class_Num
Digital	Device_ID (PK), *Resource_ID* (FK *to Resources table*), Device_Type, Device_Model_Number, Device_Manufacturer

To create the Relational Schema, we derive the entities, attributes and relationships depicted in our ER diagram and translate them into tables. At this stage in the process, it is imperative to declare all primary and foreign keys. Each entity in the diagram becomes a table in the relational schema, and relationships are implemented using foreign keys to enforce referential integrity. Our justification for the foreign key relationships is as follows:

Attribute	***Table used in***	***References***	***Justification***
User_ID	**Loans**	*Users*	- A loan cannot exist without a valid User - The database can easily retrieve all loans associated with a specific user
Copy_Num, Resource_ID	**Loans**	*Copies*	- Connects each loan to a specific copy of a resource - Tied to valid copies listed in the Copies table
Loan_ID	**Fines**	*Loans*	- Each fine is tied to a specific loan
Resource_ID	**Copies**	*Resources*	- Each copy belongs to a specific resource (eg. multiple copies of the same book or device)
Resource_ID	**Books** **Devices**	*Resources*	- Links specialised entities (Books and Digital Devices) to general entity (Resources) - Books and Devices inherit attributes from Resources, while also having their own unique attributes
User_ID	**Reservations**	*Users*	- Connects a reservation to a specific User - Only valid Users can make reservations
Resource_ID	**Reservations**	*Copies*	- Links reservations to specific copies of a resource - Reservations can only be made for available copies

1.4 Normalisation [2]

Universal attributes – *Universal relation that includes all attributes*

Universal relations: (User_ID, User_Full_Name, User_Email, User_Home_Address, User_Type, User_Account_Status, Loan_ID, Loan_Date, Loan_Due_Date, Loan_Return_Date, Copy_Num, Resource_ID, Resource_Location, Fine_ID, Fine_Amt, Fine_Status, Fine_Date, Resource_Loan_Pd, Book_ID, Book_Format, Book_Author, Book_Title, Book_ISBN, Book_Class_Num, Device_ID, Device_Type, Device_Model_Number, Device_Manufacturer, Device_Serial_Num, Copy_Avail_Stat, Reservation_ID, Reserv_Date, Reserv_Notif_Date, Reserv_Expir_Date, Reserv_Decline_Count)

1NF – *Ensure all attributes contain atomic values without repeating groups.*

We separate data into distinct tables defined by primary keys, ensuring each piece of information is stored only once. Multi-valued attributes are split into separate tables (e.g. Copies table to track individual instances of a resource rather than storing multiple copies as a repeated group in the Resources table). We eliminate any multi-valued attributes to guarantee our design adheres to 1NF:

Users
(User_ID (PK), User_Full_Name, User_Email, User_Home_Address, User_Type, User_Account_Status)

Loans
(Loan_ID (PK), Loan_Date, Loan_Due_Date, Loan_Return_Date, User_ID, Copy_Num (FK), Resource_ID)

Fines
(Fine_ID (PK), Loan_ID, Fine_Amt, Fine_Status, Fine_Date)

Resources
(Resource_ID (PK), Resource_Loan_Pd, Resource_Location Book_Format, Book_Author, Book_Title, Book_ISBN, Book_Class_Num, Device_Type, Device_Model_Number, Device_Manufacturer)

Copies
(Copy_Num (PK), Resource_ID (PK), Copy_Avail_Stat, Device_Serial_Num)

Reservations
(Reservation_ID (PK), Reserv_Date, Reserv_Notif_Date, Reserv_Expir_Date, Reserv_Decline_Count, User_ID, Copy_Num, Resource_ID

2NF *– Eliminate partial dependencies.*

In our 1NF representation, the Resources table has a partial dependency that needs fixing. Attributes related to books (like author and title) and devices (like model number) depend on the Resource_ID type. It is worth repeating here that in our system, resource type is represented as a view, so as not to duplicate data. Books & Devices are separated to avoid impractically storing multiple NULL values in the Resources table for either book or device attributes. Therefore, the Resource table is now split into three tables, each with its own primary keys, representing its logical metadata. The 2NF design is as follows:

Users
(User_ID (PK), User_Full_Name, User_Email, User_Home_Address, User_Type, User_Account_Status)

Loans
(Loan_ID (PK), Loan_Date, Loan_Due_Date, Loan_Return_Date, User_ID, Copy_Num (FK), Resource_ID (FK))

Fines
(Fine_ID (PK), Loan_ID, Fine_Amt, Fine_Status, Fine_Date)

Resources
(Resource_ID (PK), Resource_Loan_Pd)

Books
(Book_ID (PK), Resource_ID, Book_Format, Book_Author, Book_Title, Book_ISBN, Book_Class_Num)

Digital Devices
(Device_ID (PK), Resource_ID, Device_Type, Device_Model_Number, Device_Manufacturer)

Copies
(Copy_Num (PK), Resource_ID (PK), Copy_Avail_Stat, Device_Serial_Num)

Reservations
(Reservation_ID (PK), Reserv_Date, Reserv_Notif_Date, Reserv_Expir_Date, Reserv_Decline_Count, User_ID, Copy_Num, Resource_ID)

Each relation has a unique primary key, and all non-key attributes are fully dependent on this primary key. The above tables now satisfy the conditions for 2NF.

3NF *– Eliminate transitive dependencies.*

Now that our design is in 2NF, we need to eliminate any transitive dependencies. This involves ensuring all attributes fully depend on the key, the whole key, and nothing but the key. Looking specifically at the non-prime attributes in our design, it does not appear to have any transitive dependencies. We have avoided this in the following ways:

- Housing data in the relevant tables – Non-prime attributes are kept in their tables and referenced via primary and foreign keys
- Foreign keys manage relationships without including additional dependent attributes
- No redundant or derived attributes – Redundant attributes are avoided by deriving data through relationships

Final 3NF design

Users
(User_ID (PK), User_Full_Name, User_Email, User_Home_Address, User_Type, User_Account_Status)

Loans
(Loan_ID (PK), Loan_Date, Loan_Due_Date, Loan_Return_Date, User_ID (FK), Copy_Num (FK), Resource_ID (FK))

Fines
(Fine_ID (PK), Loan_ID (FK), Fine_Amt, Fine_Status, Fine_Date)

Resources
(Resource_ID (PK), Resource_Loan_Pd, Resource_Location)

Books
(Book_ID (PK), Resource_ID (FK), Book_Format, Book_Author, Book_Title, Book_ISBN, Book_Class_Num)

Digital Devices
(Device_ID (PK), Resource_ID (FK), Device_Type, Device_Model_Number, Device_Manufacturer)

Copies
(Copy_Num (PK), Resource_ID (PK & FK), Copy_Avail_Stat, Device_Serial_Num)

Reservations
(Reservation_ID (PK), Reserv_Date, Reserv_Notif_Date, Reserv_Expir_Date, Reserv_Decline_Count, User_ID (FK), Copy_Num (FK), Resource_ID(FK)

Part 2: Database Implementation (Oracle LiveSQL)

2.1 Creating the Tables

NOTE: Some DROP TABLE commands have been removed to save space. Only two have been placed for illustration.

-- Drop the users table if it exists
```
DROP TABLE users CASCADE CONSTRAINTS;
```

-- Create the users table
```
CREATE TABLE users (
    user_id NUMBER(9, 0) PRIMARY KEY,
    user_full_name VARCHAR2(100) NOT NULL,
    user_email VARCHAR2(100) UNIQUE NOT NULL,
    user_home_address VARCHAR2(255),
        user_type VARCHAR2(10) CHECK (user_type IN
('Student', 'Staff')),
            user_account_status VARCHAR2(10) CHECK
(user_account_status IN ('Active', 'Suspended')),
    loan_limit NUMBER NOT NULL -- Loan limit determined
by user_type
);
```

-- Set loan limit for ease of insertion of dummy data [3][4]
```
CREATE OR REPLACE TRIGGER set_default_loan_limit
BEFORE INSERT ON users
FOR EACH ROW
BEGIN
    -- Dynamically set loan_limit based on user_type
    IF :NEW.loan_limit IS NULL THEN
        IF :NEW.user_type = 'Student' THEN
            :NEW.loan_limit := 5;
        ELSIF :NEW.user_type = 'Staff' THEN
            :NEW.loan_limit := 10;
        ELSE
            :NEW.loan_limit := 0; -- Default fallback
(optional, for unexpected user types)
        END IF;
    END IF;
END;
/
```

-- Drop the resources table if it exists
```
DROP TABLE resources CASCADE CONSTRAINTS;
```
-- From hereon, DROP TABLE is removed from here to save space. It's still in script.

-- Create the resources table
```
CREATE TABLE resources (
    resource_id NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
    location VARCHAR2(100),
    loan_period NUMBER(5)
);
```

-- Create the Devices table
```
CREATE TABLE devices (
     device_id NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
                    resource_id     NUMBER     REFERENCES
resources(resource_id) ON DELETE CASCADE,
    manufacturer VARCHAR2(100),
    model_number VARCHAR2(100),
    device_type VARCHAR2(50) NOT NULL
);
```

-- Create the Books table
```
CREATE TABLE books (
      book_id NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
                    resource_id     NUMBER     REFERENCES
resources(resource_id) ON DELETE CASCADE,
    title VARCHAR2(255) NOT NULL,
    author VARCHAR2(255) NOT NULL,
    class_number NUMBER,
    isbn VARCHAR2(20) UNIQUE,
       book_format VARCHAR2(50) CHECK (book_format IN
('Physical Book', 'Ebook'))
);
```

-- Create the Copies table
```
CREATE TABLE copies (
                    resource_id     NUMBER     REFERENCES
resources(resource_id) ON DELETE CASCADE,
    copy_num NUMBER,
    serial_number VARCHAR2(50) UNIQUE,
          availability_status   VARCHAR2(10)   CHECK
(availability_status IN ('Available', 'Loaned')),
    PRIMARY KEY (resource_id, copy_num)
);
```

-- Create the Reservations table
```
CREATE TABLE reservations (
      reservation_id NUMBER GENERATED BY DEFAULT AS
IDENTITY PRIMARY KEY,
    user_id NUMBER REFERENCES users(user_id) ON DELETE
CASCADE,
                    resource_id     NUMBER     REFERENCES
resources(resource_id) ON DELETE CASCADE,
    reserv_date DATE DEFAULT SYSDATE,
    reserv_notif_date DATE,
    reserv_expir_date DATE,
        reserv_decline_count NUMBER DEFAULT 0 CHECK
(reserv_decline_count <= 3)
);
```

-- Create the Loans table
```
CREATE TABLE loans (
     loan_id NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
    loan_date DATE NOT NULL,
    loan_due_date DATE,
    user_id NUMBER REFERENCES users(user_id) ON DELETE
CASCADE,
    resource_id NUMBER NOT NULL,
    copy_num NUMBER NOT NULL,
    loan_return_date DATE,
       FOREIGN KEY (resource_id, copy_num) REFERENCES
copies(resource_id, copy_num) ON DELETE CASCADE
);
```

-- Created a unique index for loans to ensure that no 2 copies of the same resource taken out at the same time
```
CREATE UNIQUE INDEX unique_active_loans_idx
ON loans (
    CASE
        WHEN loan_return_date IS NULL THEN resource_id
ELSE NULL END,
    CASE
        WHEN loan_return_date IS NULL THEN copy_num
ELSE NULL END
);
```

-- Create the fines table
```
CREATE TABLE fines (
     fine_id NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
    loan_id NUMBER REFERENCES loans(loan_id) ON DELETE
CASCADE,
    user_id NUMBER REFERENCES users(user_id) ON DELETE
CASCADE,
    fine_amt NUMBER,
       fine_status VARCHAR2(10) CHECK (fine_status IN
('PAID', 'NOT PAID')),
    fine_date DATE);
```

2.2 Populating the Tables

*NOTE: We have a **comprehensive** number of INSERT INTO statements in the SQL script. INSERT INTO statements here have been **reduced** to the first 3 and last 3 inserts to ensure readability while providing sufficient samples.*

2.2.1 INSERT INTO **USERS**

```
INSERT INTO users (user_id, user_full_name, user_email, user_home_address, user_type, user_account_status) VALUES
(202331330, 'Rose Gamble', 'rose.gamble@adp.ac.uk', '84 Oak Drive, London, SE14 5XS', 'Student', 'Active');
INSERT INTO users (user_id, user_full_name, user_email, user_home_address, user_type, user_account_status) VALUES
(202285795, 'Timothy Frank', 'timothy.frank@adp.ac.uk', '53 Magnolia Way, London, SW12 1UR', 'Student', 'Active');
INSERT INTO users (user_id, user_full_name, user_email, user_home_address, user_type, user_account_status) VALUES
(202343920, 'Steven Ellis', 'steven.ellis@adp.ac.uk', '26 Rose Lane, London, NW10 7PL', 'Student', 'Active');
```

*-- **44 INSERT INTO USERS STATEMENTS REMOVED** FROM THIS DOCUMENT FOR READABILITY --*

```
INSERT INTO users (user_id, user_full_name, user_email, user_home_address, user_type, user_account_status) VALUES
(202224759, 'Carter Peterson', 'carter.peterson@adp.ac.uk', '38 Maple Road, London, SE24 8FG', 'Staff', 'Active');
INSERT INTO users (user_id, user_full_name, user_email, user_home_address, user_type, user_account_status) VALUES
(200263880, 'Sofia Gray', 'sofia.gray@adp.ac.uk', '46 Birch Avenue, London, SW19 5LG', 'Staff', 'Active');
INSERT INTO users (user_id, user_full_name, user_email, user_home_address, user_type, user_account_status) VALUES
(202147977, 'Owen Wood', 'owen.wood@adp.ac.uk', '19 Walnut Close, London, NW5 7QP', 'Staff', 'Active');
```

-- Queries to view Student and Staff after insertion for verification

```
SELECT * FROM users WHERE user_type = 'Student';
SELECT * FROM users WHERE user_type = 'Staff';
```

-- Dropped Trigger to avoid mutating tables. Trigger only created for insertion of dummy data.

```
DROP TRIGGER set_default_loan_limit;
```

2.2.2 INSERT INTO **RESOURCES**

```
INSERT INTO resources (resource_id, location, loan_period) VALUES (1, 'X-000', 21);
INSERT INTO resources (resource_id, location, loan_period) VALUES (2, '0-072', 21);
INSERT INTO resources (resource_id, location, loan_period) VALUES (3, '2-018', 21);
```

*-- **19 INSERT INTO RESOURCES STATEMENTS REMOVED** FROM THIS DOCUMENT FOR READABILITY --*

```
INSERT INTO resources (resource_id, location, loan_period) VALUES (23, '0-063', 3);
INSERT INTO resources (resource_id, location, loan_period) VALUES (24, '1-066', 3);
INSERT INTO resources (resource_id, location, loan_period) VALUES (25, '0-093', 3);
```

2.2.3 INSERT INTO **BOOKS**

```
INSERT INTO books (book_id, resource_id, title, author, class_number, isbn, book_format) VALUES (1, 1, 'The Quantum
Enigma', 'Dr. Alice Hawthorne', 530, '978-3-32-442443-6', 'Ebook');
INSERT INTO books (book_id, resource_id, title, author, class_number, isbn, book_format) VALUES (2, 2, 'Culinary
Adventures', 'Chef Marco Rossi', 641, '978-1-72-068043-6', 'Physical Book');
INSERT INTO books (book_id, resource_id, title, author, class_number, isbn, book_format) VALUES (3, 3, 'History of the
Renaissance', 'Dr. Peter Campbell', 940, '978-8-35-246400-6', 'Physical Book');
```

*-- **12 INSERT INTO BOOKS STATEMENTS REMOVED** FROM THIS DOCUMENT FOR READABILITY –*

```
INSERT INTO books (book_id, resource_id, title, author, class_number, isbn, book_format) VALUES (16, 16, 'Modern
Architecture', 'Anna Ferguson', 720, '978-7-74-791701-2', 'Ebook');
INSERT INTO books (book_id, resource_id, title, author, class_number, isbn, book_format) VALUES (17, 17, 'Wildlife of
North America', 'Jake Roberts', 599, '978-9-28-340413-3', 'Physical Book');
INSERT INTO books (book_id, resource_id, title, author, class_number, isbn, book_format) VALUES (18, 18, 'Sociology in
the Modern Age', 'Dr. Elaine Voss', 301, '978-9-27-620798-7', 'Physical Book');
```

2.2.4 INSERT INTO **DEVICES**

```
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (1, 19, 'ALIENWARE', 'M16
R2', 'LAPTOP');
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (2, 20, 'HP', 'Dragonfly
G4', 'LAPTOP');
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (3, 21, 'KINDLE',
'Paperwhite', 'E-READER');
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (4, 22, 'SAMSUNG', 'Galaxy
Tab S8', 'TABLET');
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (5, 23, 'DELL', 'XPS 13',
'LAPTOP');
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (6, 24, 'APPLE', 'iPad
Pro', 'TABLET');
INSERT INTO devices (device_id, resource_id, manufacturer, model_number, device_type) VALUES (7, 25, 'KOBO', 'Clara HD',
'E-READER');
```

2.2.5 INSERT INTO **COPIES**

```
INSERT INTO copies (resource_id, copy_num, serial_number, availability_status) VALUES (1, 1, NULL, 'Loaned');
INSERT INTO copies (resource_id, copy_num, serial_number, availability_status) VALUES (1, 2, NULL, 'Loaned');
INSERT INTO copies (resource_id, copy_num, serial_number, availability_status) VALUES (1, 3, NULL, 'Loaned');
```

```
-- 74 INSERT INTO COPIES STATEMENTS REMOVED FROM THIS DOCUMENT FOR READABILITY –
INSERT INTO copies (resource_id, copy_num, serial_number, availability_status) VALUES (24, 3, 'J0D4CXJU2269', 'Loaned');
INSERT INTO copies (resource_id, copy_num, serial_number, availability_status) VALUES (25, 1, 'KO8607AP', 'Available');
INSERT INTO copies (resource_id, copy_num, serial_number, availability_status) VALUES (25, 2, 'KO5908LO', 'Available');

-- Query to view total number of copies we generated (This prints 80)
SELECT COUNT(*) FROM copies;
```

2.2.6 INSERT INTO **LOANS**

```
INSERT INTO loans (loan_date, user_id, resource_id, copy_num, loan_return_date) VALUES (TO_DATE('2024-08-07',
'YYYY-MM-DD'), '199027462', '21', '1', TO_DATE('2024-08-08', 'YYYY-MM-DD'));
INSERT INTO loans (loan_date, user_id, resource_id, copy_num, loan_return_date) VALUES (TO_DATE('2024-09-12',
'YYYY-MM-DD'), '199151978', '20', '1', TO_DATE('2024-09-20', 'YYYY-MM-DD'));
INSERT INTO loans (loan_date, user_id, resource_id, copy_num, loan_return_date) VALUES (TO_DATE('2024-09-12',
'YYYY-MM-DD'), '200248712', '25', '1', TO_DATE('2024-09-14', 'YYYY-MM-DD'))
```

```
-- 52 INSERT INTO LOANS STATEMENTS REMOVED FROM THIS DOCUMENT FOR READABILITY –
INSERT INTO loans (loan_date, user_id, resource_id, copy_num, loan_return_date) VALUES (TO_DATE('2024-11-22',
'YYYY-MM-DD'), '202211233', '21', '2', NULL);
INSERT INTO loans (loan_date, user_id, resource_id, copy_num, loan_return_date) VALUES (TO_DATE('2024-11-22',
'YYYY-MM-DD'), '202335552', '22', '3', TO_DATE('2024-11-25', 'YYYY-MM-DD'));
INSERT INTO loans (loan_date, user_id, resource_id, copy_num, loan_return_date) VALUES (TO_DATE('2024-11-22',
'YYYY-MM-DD'), '202321415', '24', '3', NULL);
```

2.2.7 INSERT INTO **RESERVATIONS**

```
INSERT INTO reservations (user_id, resource_id, reserv_date, reserv_notif_date, reserv_expir_date, reserv_decline_count)
VALUES (202447890, 6, TO_DATE('2024-11-14', 'YYYY-MM-DD'), NULL, NULL, 0);
INSERT INTO reservations (user_id, resource_id, reserv_date, reserv_notif_date, reserv_expir_date, reserv_decline_count)
VALUES (200675782, 3, TO_DATE('2024-11-21', 'YYYY-MM-DD'), NULL, NULL, 0);
INSERT INTO reservations (user_id, resource_id, reserv_date, reserv_notif_date, reserv_expir_date, reserv_decline_count)
VALUES (202136290, 3, TO_DATE('2024-11-22', 'YYYY-MM-DD'), NULL, NULL, 0);
INSERT INTO reservations (user_id, resource_id, reserv_date, reserv_notif_date, reserv_expir_date, reserv_decline_count)
VALUES (202467706, 3, TO_DATE('2024-11-23', 'YYYY-MM-DD'), NULL, NULL, 0);
INSERT INTO reservations (user_id, resource_id, reserv_date, reserv_notif_date, reserv_expir_date, reserv_decline_count)
VALUES (202460717, 6, TO_DATE('2024-11-23', 'YYYY-MM-DD'), NULL, NULL, 0);
INSERT INTO reservations (user_id, resource_id, reserv_date, reserv_notif_date, reserv_expir_date, reserv_decline_count)
VALUES (199597776, 12, SYSDATE, NULL, NULL, 0);
```

2.2.8 INSERT INTO **FINES**

```
-- Step 1: A FUNCTION WAS CREATED TO CALCULATE THE FINE AMOUNT. This allows us to automate the amount, reducing
errors in entry. [5][6][7]
CREATE OR REPLACE FUNCTION calculate_fine(p_loan_id IN NUMBER) RETURN NUMBER IS
    resource_type VARCHAR2(20);
    resource_subtype VARCHAR2(20);
    loan_due_date DATE;
    loan_return_date DATE;
    fine_days NUMBER;
    max_fine NUMBER;
BEGIN
    -- Qualifiers needed for calculating the fines
    SELECT rtv.resource_type, rtv.resource_subtype, l.loan_due_date, l.loan_return_date
    INTO resource_type, resource_subtype, loan_due_date, loan_return_date
    FROM loans l JOIN resource_type_view rtv ON l.resource_id = rtv.resource_id
    WHERE l.loan_id = p_loan_id;
    -- Determine the max fine cap based on resource_type and resource_subtype (as in Assumptions)
    IF loan_return_date IS NOT NULL THEN
        -- Case 1: Late return
        -- Use ceiling (days are never partially computed) and GREATEST (like using MAX in Python)
        fine_days := CEIL(GREATEST(loan_return_date - loan_due_date, 0));
        CASE
            WHEN resource_type = 'Book' AND resource_subtype = 'Physical Book' THEN max_fine := 50;
            WHEN resource_type = 'Book' AND resource_subtype = 'Ebook' THEN max_fine := 50;
            WHEN resource_type = 'Device' AND resource_subtype = 'E-READER' THEN max_fine := 50;
            WHEN resource_type = 'Device' AND resource_subtype = 'TABLET' THEN max_fine := 150;
            WHEN resource_type = 'Device' AND resource_subtype = 'LAPTOP' THEN max_fine := 250;
            ELSE max_fine := 50; -- Default for unexpected cases
        END CASE;
    ELSE
        -- Case 2: Lost/Damaged
        fine_days := CEIL(GREATEST(SYSDATE - loan_due_date, 0));
        CASE
            WHEN resource_type = 'Book' AND resource_subtype = 'Physical Book' THEN max_fine := 100;
            WHEN resource_type = 'Book' AND resource_subtype = 'Ebook' THEN max_fine := 100;
            WHEN resource_type = 'Device' AND resource_subtype = 'E-READER' THEN max_fine := 100;
            WHEN resource_type = 'Device' AND resource_subtype = 'TABLET' THEN max_fine := 300;
            WHEN resource_type = 'Device' AND resource_subtype = 'LAPTOP' THEN max_fine := 500;
            ELSE max_fine := 100; -- Default for unexpected cases
        END CASE;
    END IF;
    -- Return fine amount, capped to the max allowed by using LEAST function
    RETURN LEAST(fine_days, max_fine);
END;
```

```
-- Step 2: BATCH PROCESS THE FINE INSERTION USING FUNCTION (Because FINES is a weak entity)
INSERT INTO fines (loan_id, user_id, fine_amt, fine_status, fine_date)
SELECT  l.loan_id, l.user_id, calculate_fine(l.loan_id),
    CASE
        WHEN l.loan_return_date IS NOT NULL THEN 'PAID' ELSE 'NOT PAID'
    END AS fine_status,
    CASE
        WHEN l.loan_return_date IS NOT NULL THEN l.loan_return_date ELSE SYSDATE
    END AS fine_date
FROM loans l
WHERE
    (l.loan_return_date > l.loan_due_date OR (l.loan_return_date IS NULL AND SYSDATE > l.loan_due_date))
    AND NOT EXISTS (
        SELECT 1 FROM fines f WHERE f.loan_id = l.loan_id)
```

2.3 Creating Views

2.3.1 — VIEW BOOK AVAILABILITY

This view pulls from the Book and Copies tables to make it more convenient to view how many copies of each book are currently available to be loaned and how many are currently being loaned by other users.

```
CREATE OR REPLACE VIEW book_availability AS
SELECT
    r.resource_id,
    b.title AS book_title,
    b.author AS book_author,
    b.isbn AS book_isbn,
    b.book_format AS book_format,
    SUM(CASE WHEN c.availability_status =
'Available' THEN 1 ELSE 0 END) AS
available_copies,
    SUM(CASE WHEN c.availability_status =
'Loaned' THEN 1 ELSE 0 END) AS unavailable_copies
FROM resources r
INNER JOIN books b ON r.resource_id =
b.resource_id
LEFT JOIN copies c ON r.resource_id =
c.resource_id
GROUP   BY   r.resource_id,   b.title,   b.author,
b.isbn, b.book_format;
```

RESOURCE_ID	BOOK_ID	BOOK_TITLE	BOOK_AUTHOR	BOOK_ISBN	BOOK_FORMAT	AVAILABLE_ COPIES	UNAVAILABLE_ COPIES
11	11	Astrophysics Essentials	Dr. Carla Jensen	978-6-06-037118-8	Physical Book	4	2
15	15	Advanced Microbiology	Dr. Jane Peterson	978-6-43-986967-6	Physical Book	3	0
13	13	Fundamentals of Chemistry	Prof. Leah Gupta	978-5-58-443714-3	Ebook	4	0
14	14	American Politics Today	Dr. Stephen Marks	978-5-06-289767-9	Physical Book	0	3
16	16	Modern Architecture	Anna Ferguson	978-7-74-791701-2	Ebook	2	1
1	1	The Quantum Enigma	Dr. Alice Hawthorne	978-3-32-442443-6	Ebook	1	3
2	2	Culinary Adventures	Chef Marco Rossi	978-1-72-068043-6	Physical Book	4	1
4	4	Introduction to Psychology	Dr. Lena Ortega	978-2-57-081911-2	Ebook	3	1
9	9	French Grammar Simplified	Marie Dubois	978-7-13-908910-4	Physical Book	3	3
7	7	The Solar System Explained	Dr. Emily Wu	978-7-26-155231-3	Ebook	1	1
17	17	Wildlife of North America	Jake Roberts	978-9-28-340413-3	Physical Book	3	1
6	6	The Art of Meditation	Sarah Ling	978-4-22-116924-0	Physical Book	0	2
8	8	Ecology and Environment	Dr. Robert Yang	978-7-27-172716-9	Physical Book	2	1
10	10	The Poetry of Nature	William J. Roberts	978-1-59-968165-0	Ebook	1	0
18	18	Sociology in the Modern Age	Dr. Elaine Voss	978-9-27-620798-7	Physical Book	1	2
3	3	History of the Renaissance	Dr. Peter Campbell	978-8-35-246400-6	Physical Book	1	4
5	5	Algorithms and Data Structures	John Tanaka	978-6-31-363733-8	Physical Book	3	0
12	12	Exploring Ancient Civilizations	Dr. Mike Henders	978-2-39-316881-1	Physical Book	0	1

2.3.2 – VIEW DEVICE AVAILABILITY

This view functions just like the Book Availability view but for the library's Devices.

```
CREATE OR REPLACE VIEW device_availability AS
SELECT
    r.resource_id,
    d.manufacturer AS device_manufacturer,
    d.model_number AS device_model_number,
    d.device_type AS device_type,
    SUM(CASE WHEN c.availability_status = 'Available'
THEN 1 ELSE 0 END) AS available_copies,
    SUM(CASE WHEN c.availability_status = 'Loaned' THEN
1 ELSE 0 END) AS unavailable_copies
FROM resources r
INNER JOIN devices d ON r.resource_id = d.resource_id
LEFT JOIN copies c ON r.resource_id = c.resource_id
GROUP BY r.resource_id, d.manufacturer, d.model_number,
d.device_type;
```

RESOURCE_ID	DEVICE_MANUFACTURER	DEVICE_MODEL_NUMBER	DEVICE_TYPE	AVAILABLE_COPIES	UNAVAILABLE_COPIES
20	HP	Dragonfly G4	LAPTOP	1	1
21	KINDLE	Paperwhite	E-READER	1	2
23	DELL	XPS 13	LAPTOP	1	2
22	SAMSUNG	Galaxy Tab S8	TABLET	1	2
24	APPLE	iPad Pro	TABLET	1	2
25	KOBO	Clara HD	E-READER	2	0
19	ALIENWARE	M16 R2	LAPTOP	1	1

2.3.3 — VIEW RESOURCE TYPES

This view displays all the resources in the library, categorized by their respective types. This not only offers a more elegant and user-friendly naming convention for the devices, but it also simplifies the process of querying the system. Users can easily locate both books and devices.

```
CREATE OR REPLACE VIEW resource_type_view AS
SELECT
    RESOURCE_ID,
    TITLE AS NAME,
    'Book' AS resource_type,
    BOOK_FORMAT AS resource_subtype
FROM
    Books
UNION ALL
SELECT
    RESOURCE_ID,
    MANUFACTURER || ' ' || MODEL_NUMBER AS NAME,
    'Device' AS resource_type,
    DEVICE_TYPE AS resource_subtype
FROM
    Devices;
```

RESOURCE_ID	NAME	RESOURCE_TYPE	RESOURCE_SUBTYPE
1	The Quantum Enigma	Book	Ebook
2	Culinary Adventures	Book	Physical Book
3	History of the Renaissance	Book	Physical Book
4	Introduction to Psychology	Book	Ebook
5	Algorithms and Data Structures	Book	Physical Book
6	The Art of Meditation	Book	Physical Book
7	The Solar System Explained	Book	Ebook
8	Ecology and Environment	Book	Physical Book
9	French Grammar Simplified	Book	Physical Book
10	The Poetry of Nature	Book	Ebook
11	Astrophysics Essentials	Book	Physical Book
12	Exploring Ancient Civilizations	Book	Physical Book
13	Fundamentals of Chemistry	Book	Ebook
14	American Politics Today	Book	Physical Book
15	Advanced Microbiology	Book	Physical Book
16	Modern Architecture	Book	Ebook
17	Wildlife of North America	Book	Physical Book
18	Sociology in the Modern Age	Book	Physical Book
19	ALIENWARE M16 R2	Device	LAPTOP
20	HP Dragonfly G4	Device	LAPTOP
21	KINDLE Paperwhite	Device	E-READER
22	SAMSUNG Galaxy Tab S8	Device	TABLET
23	DELL XPS 13	Device	LAPTOP
24	APPLE iPad Pro	Device	TABLET
25	KOBO Clara HD	Device	E-READER

2.3.4 – VIEW USERS' TOTAL LOAN COUNTS

This view offers a convenient way to view how many loans each user has ever taken out in the library, in descending order.

```
CREATE OR REPLACE VIEW user_loans_count AS
SELECT
    u.user_id,
    u.user_full_name,
    u.user_type,
    COUNT(l.loan_id) AS total_loans
FROM
    users u
LEFT JOIN
    loans l ON u.user_id = l.user_id
GROUP BY
    u.user_id, u.user_full_name, u.user_type
ORDER BY
    total_loans DESC;
```

(View user total counts continued)

USER_ID	USER_FULL_NAME	USER_TYPE	TOTAL_LOANS
202424207	Emily Johnson	Student	5
202289533	Anthony Phillips	Staff	4
202254578	Logan Allen	Student	4
199027462	Ellie Sanders	Staff	3
202321415	Daniel Turner	Student	3
200263880	Sofia Gray	Staff	3
202415742	Jack Moore	Student	3
202244677	Liam King	Student	3
202267393	Mason Hall	Student	3
202367666	Amelia Martin	Student	3
202335552	Ava Mitchell	Student	3
202322746	William White	Student	2
202248681	Evelyn Thompson	Student	2
202429647	Ethan Wright	Student	2
202333711	Henry Smith	Student	2
202332267	Harper Edwards	Student	1
202346911	James Lewis	Student	1
202442989	Caleb Powell	Student	1
202221592	Olivia Harris	Student	1
199172725	Hannah Reed	Staff	1
202448446	Michael Clark	Student	1
202392700	Mia Green	Student	1
202211233	Charlotte Young	Student	1
199151978	Andrew Fisher	Staff	1
202447890	Sarah Walker	Student	1
200248712	Aria Evans	Staff	1
202437815	Benjamin Lee	Student	1
202498064	Isabella Adams	Student	1
200317178	Jackson Cook	Staff	1
202343920	Steven Ellis	Student	0
199597776	Grace Ward	Staff	0
202255506	Lily Bell	Student	0
202136290	Gabriel Foster	Staff	0
201277987	Aiden Murphy	Staff	0
202351058	Alexander Hughes	Student	0
202270833	Jacob Parker	Student	0
202271017	Emma Davis	Student	0
202331330	Rose Gamble	Student	0
200675782	Layla Bailey	Staff	0
202297777	Sophia Carter	Student	0
202387579	Lucas Scott	Student	0
202474146	Chloe Roberts	Student	0
202460717	Ella Brown	Student	0
202467706	Ryan Murphy	Staff	0
200644770	Scarlett Ross	Staff	0
202475036	Sebastian Price	Student	0
202285795	Timothy Frank	Student	0
202224759	Carter Peterson	Staff	0
202478705	Ella Simmons	Student	0
202147977	Owen Wood	Staff	0

2.4 Implementing SQL Queries

2.4.1 Simple Queries

including SELECT and WHERE

2.4.1.1 QUERY ALL USERS IN THE DATABASE WHO WERE ADDED IN 2024

Use Case: Get report of all users who joined in 2024 to track membership growth or evaluate services for new members.

```
SELECT user_id, user_full_name, user_type
FROM users
WHERE user_id BETWEEN 202400000 AND 202499999;
```

USER_ID	USER_FULL_NAME	USER_TYPE
202460717	Ella Brown	Student
202415742	Jack Moore	Student
202424207	Emily Johnson	Student
202448446	Michael Clark	Student
202447890	Sarah Walker	Student
202437815	Benjamin Lee	Student
202429647	Ethan Wright	Student
202498064	Isabella Adams	Student
202474146	Chloe Roberts	Student
202475036	Sebastian Price	Student
202478705	Ella Simmons	Student
202442989	Caleb Powell	Student
202467706	Ryan Murphy	Staff

2.4.1.2 QUERY ALL RESERVATIONS OF A SPECIFIC USER

Use Case: Allows to check the reservation history of a specific user who claims they haven't received notifications for their reserved books.

```
SELECT user_id, reservation_id, resource_id,
reserv_date
FROM reservations
WHERE user_id = 202460717;
```

USER_ID	RESERVATION_ID	RESOURCE_ID	RESERV_DATE
202460717	5	6	23-NOV-24

2.4.1.3 QUERY ALL BOOKS ABOUT A PARTICULAR SUBJECT

(for example: Biology)
Use Case: A student/ librarian can use class_number to fetch books within the relevant Dewey Decimal range.

```
SELECT book_id, title, author, class_number,
isbn, book_format
FROM books
WHERE class_number BETWEEN 570 AND 579;
```

BOOK_ID	TITLE	AUTHOR	CLASS_NUMBER	ISBN	BOOK_FORMAT
8	Ecology and Environment	Dr. Robert Yang	577	978-7-27-172716-9	Physical Book
15	Advanced Microbiology	Dr. Jane Peterson	579	978-6-43-986967-6	Physical Book

2.4.1.4 QUERY ALL LOANED ITEMS WHICH ARE OVERDUE TO BE RETURNED

Use Case: Library wants to notify users who have overdue loans. This query identifies the overdue items and the users responsible for returning them.

```
SELECT loan_id, user_id, resource_id,
loan_due_date
FROM loans
WHERE loan_due_date < SYSDATE AND
loan_return_date IS NULL
ORDER BY loan_due_date ASC;
```

LOAN_ID	USER_ID	RESOURCE_ID	LOAN_DUE_DATE
8	202221592	9	17-OCT-24
42	202244677	22	16-NOV-24
43	200263880	21	19-NOV-24
18	202267393	1	19-NOV-24
21	202254578	6	22-NOV-24
20	202367666	3	22-NOV-24
22	202424207	11	22-NOV-24
47	202346911	23	23-NOV-24
24	202289533	8	23-NOV-24
52	199172725	24	24-NOV-24
56	202211233	21	25-NOV-24
58	202321415	24	25-NOV-24
30	202254578	16	28-NOV-24
29	202415742	14	28-NOV-24
31	202321415	9	30-NOV-24
32	200263880	18	30-NOV-24

2.4.2 Intermediate Queries

including JOIN

2.4.2.1 FIND THE DEVICES THAT ARE CURRENTLY BEING LOANED AND THEIR USERS' DETAILS

Use Case: IT department needs to track all currently loaned devices (e.g., tablets, laptops) and identify which users are responsible for returning them.

```
SELECT u.user_id, u.user_full_name, r.name as
"DEVICE", l.loan_date, l.loan_due_date
FROM resource_type_view r
JOIN loans l ON r.resource_id = l.resource_id
JOIN users u ON l.user_id = u.user_id
WHERE l.loan_return_date IS NULL AND
r.resource_type = 'Device';
```

USER_ID	USER_FULL_NAME	DEVICE	LOAN_DATE	LOAN_DUE_DATE
202244677	Liam King	SAMSUNG Galaxy Tab S8	13-NOV-24	16-NOV-24
202346911	James Lewis	DELL XPS 13	20-NOV-24	23-NOV-24
202211233	Charlotte Young	KINDLE Paperwhite	22-NOV-24	25-NOV-24
202321415	Daniel Turner	APPLE iPad Pro	22-NOV-24	25-NOV-24
199172725	Hannah Reed	APPLE iPad Pro	21-NOV-24	24-NOV-24
200263880	Sofia Gray	KINDLE Paperwhite	16-NOV-24	19-NOV-24

2.4.2.2 QUERY ALL COPIES OF PHYSICAL BOOKS THAT ARE AVAILABLE TO BE LOANED

Use Case: Librarian can check which physical books are available for users visiting the library. Helps quickly locate books on the shelf.

```
SELECT c.copy_num, b.title, c.availability_status
FROM books b
JOIN copies c ON b.resource_id = c.resource_id
WHERE b.book_format = 'Physical Book' AND
c.availability_status = 'Available';
```

COPY_NUM	TITLE	AVAILABILITY_STATUS
1	Culinary Adventures	Available
3	Culinary Adventures	Available
4	Culinary Adventures	Available
5	Culinary Adventures	Available
3	History of the Renaissance	Available
1	Algorithms and Data Structures	Available
2	Algorithms and Data Structures	Available
3	Algorithms and Data Structures	Available
2	Ecology and Environment	Available
3	Ecology and Environment	Available
1	French Grammar Simplified	Available
4	French Grammar Simplified	Available
5	French Grammar Simplified	Available
2	Astrophysics Essentials	Available
4	Astrophysics Essentials	Available
5	Astrophysics Essentials	Available
6	Astrophysics Essentials	Available
1	Advanced Microbiology	Available
2	Advanced Microbiology	Available
3	Advanced Microbiology	Available
2	Wildlife of North America	Available
3	Wildlife of North America	Available
4	Wildlife of North America	Available
1	Sociology in the Modern Age	Available

2.4.2.3 LIST ALL RESOURCES CURRENTLY LOANED BY A PARTICULAR USER

Use Case: User wants to confirm the resources they have currently borrowed. The librarian runs this query to provide an accurate summary.

```
SELECT u.user_id, u.user_full_name,
l.resource_id, r.name as "RESOURCE_NAME",
l.loan_date, l.loan_due_date
FROM loans l
JOIN users u ON l.user_id = u.user_id
JOIN resource_type_view r ON l.resource_id =
r.resource_id
WHERE l.loan_return_date IS NULL AND u.user_id =
200263880;
```

USER_ID	USER_FULL_NAME	RESOURCE_ID	RESOURCE_NAME	LOAN_DATE	LOAN_DUE_DATE
200263880	Sofia Gray	17	Wildlife of North America	22-NOV-24	13-DEC-24
200263880	Sofia Gray	18	Sociology in the Modern Age	09-NOV-24	30-NOV-24
200263880	Sofia Gray	21	KINDLE Paperwhite	16-NOV-24	19-NOV-24

2.4.2.4 FIND DETAILS OF ALL OUTSTANDING FINES AND THE USER ASSOCIATED WITH EACH ONE

Use Case: Library admin prepares a report of all unpaid fines and their associated users to ensure payments are collected or accounts are flagged for suspension at end of year.

```
SELECT u.user_id, u.user_full_name, f.fine_amt AS
fine_amount, f.fine_status
FROM Users u LEFT JOIN Fines f ON u.user_id =
f.user_id
WHERE f.fine_status = 'NOT PAID'
ORDER BY u.user_id;
```

USER_ID	USER_FULL_NAME	FINE_AMOUNT	FINE_STATUS
199172725	Hannah Reed	7	NOT PAID
200263880	Sofia Gray	12	NOT PAID
200263880	Sofia Gray	1	NOT PAID
202211233	Charlotte Young	6	NOT PAID
202221592	Olivia Harris	45	NOT PAID
202244677	Liam King	15	NOT PAID
202254578	Logan Allen	3	NOT PAID
202254578	Logan Allen	9	NOT PAID
202267393	Mason Hall	12	NOT PAID
202289533	Anthony Phillips	8	NOT PAID
202321415	Daniel Turner	1	NOT PAID
202321415	Daniel Turner	6	NOT PAID
202346911	James Lewis	8	NOT PAID
202367666	Amelia Martin	9	NOT PAID
202415742	Jack Moore	3	NOT PAID
202424207	Emily Johnson	9	NOT PAID

2.4.3 Advanced Queries

including `JOIN` and `GROUP BY`

2.4.3.1 QUERY HOW MANY ACTIVE LOANS EACH INDIVIDUAL USER HAS.

This differs from the users_loan_count view, which shows *all* loans each user has taken out rather than how many are still active.

Use Case: The library admin monitors borrowing patterns and ensures users adhere to borrowing limits. This query lists users with the highest number of active loans.

```
SELECT u.user_full_name, COUNT(l.loan_id) AS
active_loans
FROM users u
JOIN loans l ON u.user_id = l.user_id
WHERE l.loan_return_date IS NULL
GROUP BY u.user_full_name
ORDER BY active_loans DESC;
```

USER_FULL_NAME	ACTIVE_LOANS
Logan Allen	3
Emily Johnson	3
Sofia Gray	3
Liam King	3
Daniel Turner	2
Anthony Phillips	2
Amelia Martin	2
Ethan Wright	2
Jack Moore	1
William White	1
Hannah Reed	1
James Lewis	1
Isabella Adams	1
Olivia Harris	1
Evelyn Thompson	1
Ava Mitchell	1
Mason Hall	1
Charlotte Young	1
Mia Green	1
Sarah Walker	1
Benjamin Lee	1

2.4.3.2 QUERY A LIST OF USERS WITH OUTSTANDING FINES, INCLUDING THE TOTAL AMOUNT OWED, THE NUMBER OF FINES INCURRED, AND WHETHER THIS HAS RESULTED IN THEIR SUSPENSION.

Use Case: The library checks which users are suspended due to unpaid fines and calculates the total amount owed to assess financial risks and take action.

(Note that a user must owe >£10 on a single resource to be suspended, which is why Logan Allen is still an active user, but Mason Hall is suspended).

```
SELECT  u.user_id, u.user_full_name,
u.user_account_status,
        SUM(calculate_fine(l.loan_id))
        total_owed, COUNT(f.loan_id) AS
        number_of_fines
FROM users u
JOIN loans l ON u.user_id = l.user_id
JOIN fines f ON l.loan_id = f.loan_id
WHERE l.loan_return_date IS NULL
GROUP BY u.user_id, u.user_full_name,
u.user_account_status
ORDER BY u.user_account_status, total_owed desc;
```

USER_ID	USER_FULL_NAME	USER_ACCOUNT_STATUS	TOTAL_OWED	NUMBER_OF_FINES
202254578	Logan Allen	Active	12	2
202367666	Amelia Martin	Active	9	1
202424207	Emily Johnson	Active	9	1
202289533	Anthony Phillips	Active	8	1
202346911	James Lewis	Active	8	1
199172725	Hannah Reed	Active	7	1
202321415	Daniel Turner	Active	7	2
202211233	Charlotte Young	Active	6	1
202415742	Jack Moore	Active	3	1
202221592	Olivia Harris	Suspended	45	1
202244677	Liam King	Suspended	15	1
200263880	Sofia Gray	Suspended	13	2
202267393	Mason Hall	Suspended	12	1

2.4.3.3 QUERY HOW MANY TIMES EACH RESOURCE HAS BEEN LOANED, IN DESCENDING ORDER.

Use Case: At the end of the academic year, the library reviews the popularity of resources to decide whether to purchase additional copies or retire rarely used items. This allows better allocation of budget for acquiring additional copies of popular resources.

```
SELECT l.resource_id, rtv.name,
rtv.resource_subtype, COUNT(l.resource_id) AS
loan_count
FROM loans l
JOIN resource_type_view rtv ON l.resource_id =
rtv.resource_id
GROUP BY rtv.name, l.resource_id,
rtv.resource_subtype
ORDER BY loan_count DESC;
```

(Output of number of times a resource has been loaned)

RESOURCE_ID	NAME	RESOURCE_SUBTYPE	LOAN_COUNT
3	History of the Renaissance	Physical Book	6
11	Astrophysics Essentials	Physical Book	4
1	The Quantum Enigma	Ebook	4
14	American Politics Today	Physical Book	4
9	French Grammar Simplified	Physical Book	4
16	Modern Architecture	Ebook	3
23	DELL XPS 13	LAPTOP	3
17	Wildlife of North America	Physical Book	3
2	Culinary Adventures	Physical Book	3
21	KINDLE Paperwhite	E-READER	3
4	Introduction to Psychology	Ebook	2
24	APPLE iPad Pro	TABLET	2
22	SAMSUNG Galaxy Tab S8	TABLET	2
6	The Art of Meditation	Physical Book	2
20	HP Dragonfly G4	LAPTOP	2
13	Fundamentals of Chemistry	Ebook	2
18	Sociology in the Modern Age	Physical Book	2
5	Algorithms and Data Structures	Physical Book	2
8	Ecology and Environment	Physical Book	1
7	The Solar System Explained	Ebook	1
25	KOBO Clara HD	E-READER	1
12	Exploring Ancient Civilizations	Physical Book	1
19	ALIENWARE M16 R2	LAPTOP	1
10	The Poetry of Nature	Ebook	1

2.4.3.4 QUERY THE AVAILABILITY OF EACH RESOURCE AND THEIR LOCATIONS. RESOURCES ARE ORDERED BY LOCATION FOR EASIER ACCESS. THE LOCATION ‘X-000’ INDICATES THAT THE RESOURCE IS AN EBOOK AND AVAILABLE ONLINE.

Use case: User inquires about available resources &their locations in the library. Provide a detailed list of available copies and their respective physical or online locations.

```
SELECT r.resource_id, rtv.name, r.location,
COUNT(c.copy_num) AS available_copies
FROM resource_type_view rtv
JOIN resources r ON rtv.resource_id =
r.resource_id
JOIN copies c ON r.resource_id = c.resource_id
WHERE c.availability_status = 'Available'
GROUP BY r.resource_id, rtv.name, r.location
ORDER BY r.location;
```

RESOURCE_ID	NAME	LOCATION	AVAILABLE_COPIES
15	Advanced Microbiology	0-000	3
18	Sociology in the Modern Age	0-000	1
17	Wildlife of North America	0-000	3
11	Astrophysics Essentials	0-006	4
8	Ecology and Environment	0-050	2
23	DELL XPS 13	0-063	2
2	Culinary Adventures	0-072	4
9	French Grammar Simplified	0-075	3
25	KOBO Clara HD	0-093	2
22	SAMSUNG Galaxy Tab S8	1-015	2
20	HP Dragonfly G4	1-019	2
24	APPLE iPad Pro	1-066	1
19	ALIENWARE M16 R2	1-073	2
3	History of the Renaissance	2-018	1
5	Algorithms and Data Structures	2-029	3
21	KINDLE Paperwhite	2-053	1
13	Fundamentals of Chemistry	X-000	4
4	Introduction to Psychology	X-000	3
16	Modern Architecture	X-000	2
10	The Poetry of Nature	X-000	1
1	The Quantum Enigma	X-000	1
7	The Solar System Explained	X-000	1

References:

[1] M. Shekelyan. (2024). ECS740P [PowerPoint slides]

[2] “A SIMPLE GUIDE TO FIVE NORMAL FORMS IN RELATIONAL DATABASE THEORY.” Accessed: Nov. 15, 2024. [Online]. Available: https://www.roma1.infn.it/people/barone/metinf/database-normalization.pdf

[3]] Oracle, *Oracle Database PL/SQL Language Reference: CREATE TRIGGER Statement*. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/lnpls/CREATE-TRIGGER-statement.html. [Accessed: 20-Nov-2024].

[4] Oracle, Oracle Database PL/SQL Language Reference: Triggers. [Online]. Available: https://docs.oracle.com/database/121/LNPLS/triggers.htm#LNPLS2007. [Accessed: 20-Nov-2024].

[5] Oracle, Oracle Database PL/SQL Language Reference: CREATE FUNCTION Statement. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/lnpls/CREATE-FUNCTION-statement.html. [Accessed: 26-Nov-2024].

[6] Oracle Tutorial, “PL/SQL SELECT INTO Statement,” [Online]. Available: https://www.oracletutorial.com/plsql-tutorial/plsql-select-into/. [Accessed: 21-Nov-2024].

[7] Oracle Tutorial, “PL/SQL CASE Statement,” [Online]. Available: https://www.oracletutorial.com/plsql-tutorial/plsql-case-statement/. [Accessed: 21-Nov-2024].