College Library Database Management System
ECS740P - Database Systems

Group 4 – James Kelly, Adrian Bunn, Ruben Elbrond-Palmer

Introduction

The aim of this report is to outline the design of a database system for a college library and to be implemented in an Oracle database application. The following account will include the conceptual design as well as the relational mapping of entities and attributes needed for this system to be functionally implemented.

For this system to be understood before being put into practice, the following steps were taken:

- Assumptions for the system were made when creating the conceptual design
- An entity-relationship diagram (ERD) for the database to include all needed tables and their relationships
- A relational database schema to detail these relationships including their primary and foreign keys
- The normalization of the design, up to and including the 3rd normal form

Assumptions

Before diving into the design of the database, it is crucial to first establish a clear definition of its objectives and the foundational principles guiding its creation. By beginning with the system's goals, they can be distilled into the following key points:

- Loan system: facilitate the loans of eBooks, books and electronic equipment, which should include the ability to 'differentiate between members' loan limits.
- Reservation system: facilitate the reservation of eBooks, books and electronic equipment before a loan is made, including the ability to track the number of times a reservation has not been taken up resulting in reservation cancellation.
- Fines system: calculates the amount a member owes based on the number of days the loan return is overdue.

Now that the objective of the database has been declared, we can detail the starting assumptions which can be derived using the goals and description stated above:

1. There are three types of resources which can be loaned: books, eBooks and electronic equipment.
2. Resources have different loan types: 3 weeks, 3 days and some may only be used on-site.
3. Resources are located over 3 floors, separated into shelves and are categorized into class numbers which are used to identify in which subject area a particular item belongs.
4. The system will contain a record of all resources kept in the library, with the total number of copies kept in stock, including the current amount available to be loaned by a member.
5. Different membership types determine the maximum number of loans a member is allowed to make.
6. The reservation system will allow members to reserve an item to be loaned at a later date if it is currently not available.
7. Fines will be calculated based on the number of days a loan is overdue (£1 per day overdue) and will be followed by a member being suspended once the amount reaches £10.
8. A record of suspended students and their total balance owed will be kept.
9. A record of total loan count will be kept identifying popular resources and will allow library staff to increase the stock for this resource if needed.

To effectively implement these assumptions to create a conceptual database, we must go into further detail on how this library system will operate:

10. The system will be operating autonomously and will not have library staff facilitating the reservations and loans these will be done via self-checkout counters.
11. Every reservation and loan will be given a unique identification number.
12. Students may loan up to and including 5 items, and staff may loan up to and including 10 items.
13. Both students and staff reservations hold the same priority as the first one to reserve.
14. When a reservation is made a member will be notified via email once this resource becomes available.

15. If a member is unable to take up the offer of the loan or does not take up the offer within 3 days, the member with the next earliest reservation will be notified.
16. If a member is unable to take the offer of a loan after a reservation after 3 notifications, the reservation is cancelled.
17. A suspension following the fine limit will be lifted once the member pays back the full amount owed, and a record is kept of the date that this has been paid.
18. Electronic devices do not have a class number, and eBooks do not have a physical location or a class number.
19. Laptops and tablets are only to be used in the library due to their value ('*Loan type*' as on/offsite). eBook readers can be used for a maximum length of 3 days whilst all books (eBooks and physical books) can be loaned for up to 3 weeks.
20. Each set of books kept in stock are the exact same copy with the same ISBN number.
21. Some books (self-published) do not have an ISBN number which is why a 'Book ID' is used as a unique identifier.
22. The library does not digitize any physical copies itself.

E-R Diagram

After establishing and detailing our assumptions for the project, we created an entity relationship diagram to map out the entities and their relationships within the system. It provides a clear, graphical representation of the database structure allowing us to outline and understand the system's requirements.

While derived attributes are not strictly necessary in the conceptual model, we chose to include these for the sake of clarity and to illustrate the system's capabilities as they will help visualize key functionalities.

Days overdue: in the '*Loans*' entity, is calculated by comparing the current date to 'due date' which is linked to a resource's loan period.

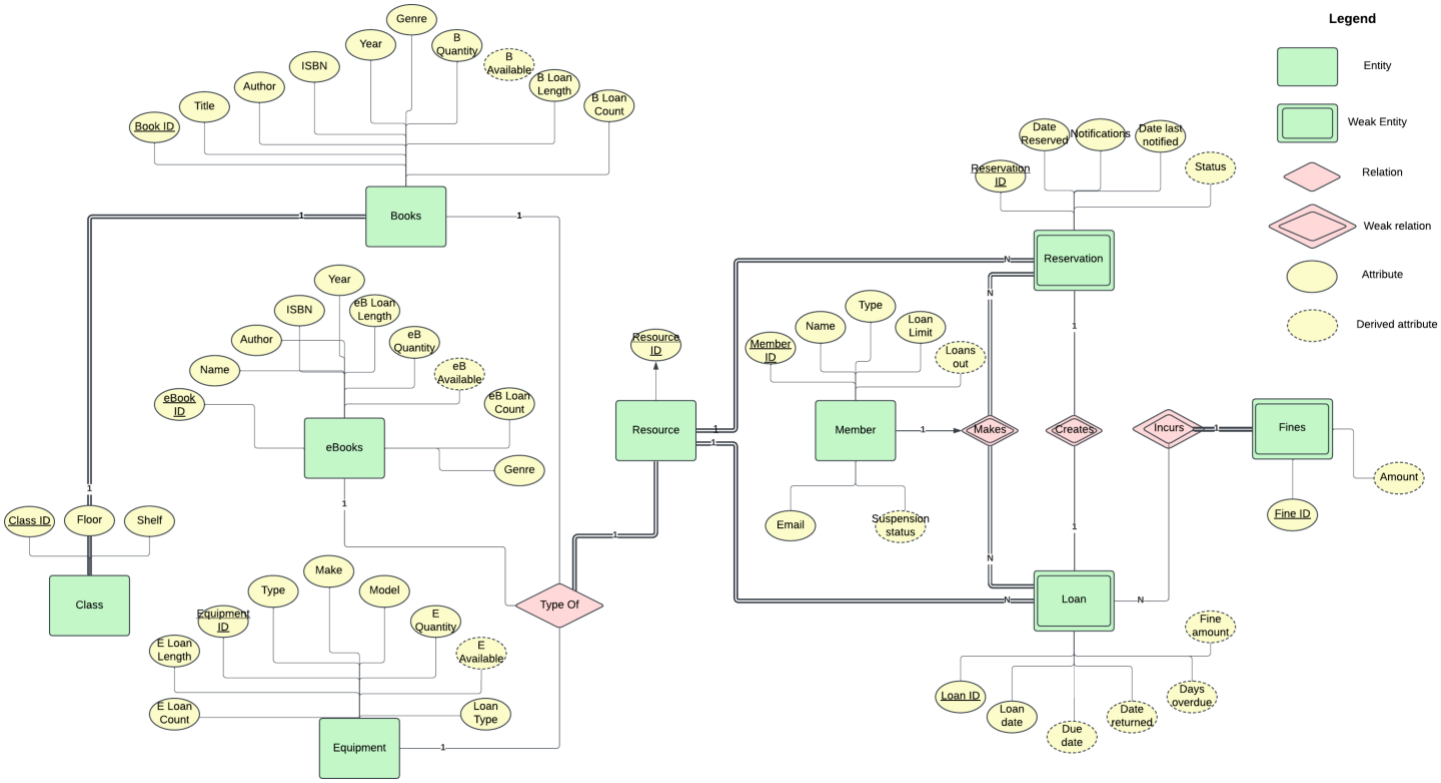
Suspension Status and *Loans Out*: in the '*Members*' entity, help track which members are currently suspended or how many items they have loaned ensuring dynamic enforcement of borrowing limits and suspensions.

Reservation Status: indicates whether a reservation is active, taken or cancelled.

Stock Availability Tracking: '*E available*' (equipment), '*eB available*' (eBooks) and '*B available*' (Books) are computed based on the total quantity in stock and the number of items currently on loan, proving real-time insight for effective resource management.

Attribute simplification: To enhance query efficiency, attributes such as '*name*', '*author*' and date attributes ('*date paid*', '*date due*', '*date returned*') are treated as atomic attributes.

The ER diagram was created using Lucidchart, effectively illustrating the system's structure, functionality and ensuring the design meets operational needs while supporting efficient data management.



Relational Schema

We are now able to clearly list and define the schema from the ER diagram. Primary and Foreign Keys will be indicated with (PK) and (FK) respectively. *Primary Keys* uniquely identify each entity, while *Foreign Keys* link related entities.

LOANS *(Loan ID(PK), Member ID, Resource ID, Loan Date, Date returned)*

Each loan is assigned a unique *Loan ID* (PK) to identify it. *Member ID* and *Resource ID* are foreign keys that link to the *Members* and *Resources* tables. *Loan Date* tracks when a resource was borrowed and *Due Date* being a derived attribute based on the resource's *Loan Length*. *Days Overdue* will be used to calculate any overdue fines.

RESERVATIONS *(Reservation ID (PK), Member ID (FK), Resource ID (FK), Notifications, Date last Notified)*

Each reservation is given a unique *Reservation ID* (PK), with *Member ID* and *Resource ID* as foreign keys. *Notifications* tracks how many times a member was notified when the resource becomes available with *Date Last Notified* recording the most recent notification date. If a member does not respond after three notifications, the reservation is cancelled which can be tracked using the *Notifications* count.

BOOKS *(Books ID(PK), ISBN, Title, Author, Genre, Year, Edition, Class ID(FK) B. Quantity, Loan Length (Days))*

Books are given the primary key *Book ID* ensuring unique identification. While *ISBN* can be used for identification where available, *Book ID* ensures uniqueness for all books. *Class ID* (FK) links to the *Class* table for categorization and *Loan Length* (days) specifies the permitted loan period for each book with *Loan Type* indicating whether a book is allowed for onsite use only.

MEMBERS *(Member ID(PK), Name, Email, Loan Limit)*

Member ID uniquely identifies each library member which is a combination of college, member type, initials, code and enrolment date. *Loan Limit* is based on membership type, dictating the maximum amount of loans to be taken out at one time (5 items for students and 10 for staff). *Suspension Status* will be derived from the *Loans* and *Fines* tables by querying any outstanding loans and fines.

EQUIPMENT *(Equip ID(PK), Type, Model, Make, E loan length, E Quantity, Loan Type)*

Equipment will be given the primary key *Equip ID* to identify it, while *Loan Type* prevents any ambiguity for items meant for onsite use only. E *Quantity* tracks the total number of identical devices kept in stock while *Loan Length* defines how long equipment can be borrowed.

FINES *(Fines ID(PK), Member ID(FK), Balance Owed)*

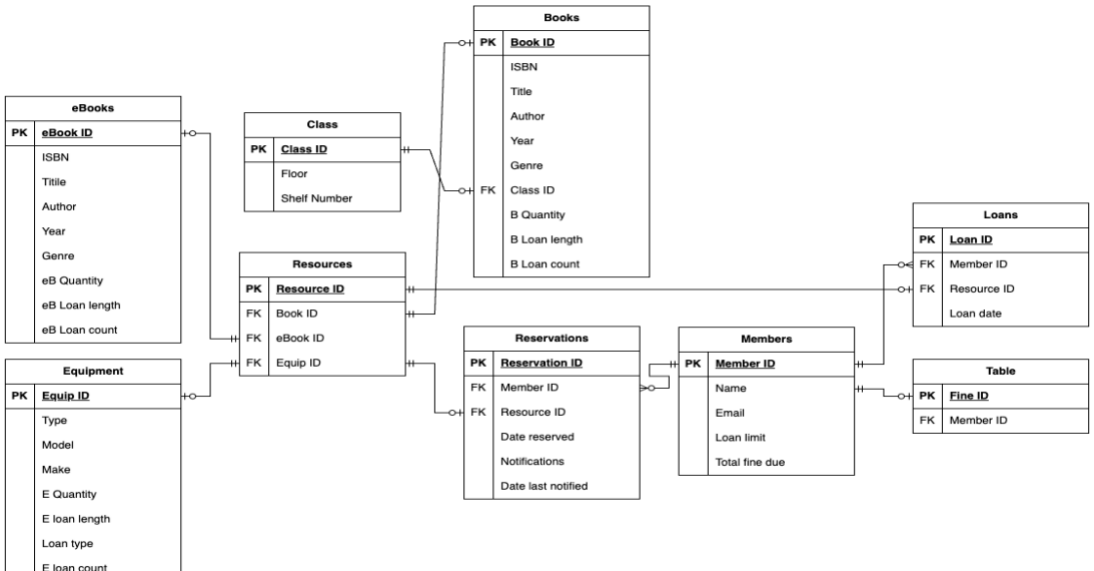
Each overdue loan generates a *Fine ID*, with *Balance Owed* being calculated by the number of overdue days by the fine rate (£1), derived from the *Loans* table.

RESOURCES *(Resource ID(PK), Book ID(FK), eBook ID(FK), Equipment ID(FK))*

Each resource (book, eBook, equipment) is assigned a unique *Resouce ID*. This simplifies references in the *Loans* table, allowing for a unifying method to track books, eBooks and equipment with a single ID, avoiding redundancy and null values with separate IDs for books, eBooks and equipment.

CLASS *(Class ID(PK), Floor, Shelf)*

Class ID as the primary key uniquely categorizes and identifies a resource's location, where *Floor* and *Shelf* are used to track physical locations for books linking them to their storage locations in the library.



Normalization

Loans ID, Member Name, Member Email, Loan Limit, Resource Type, Title, Author, Genre, ISBN, Year, Floor, Shelf, Type, Model, Make, Loan Length, Quantity, Loan Type, Loan Date, Date, Returned, Fine Amount Notifications, Date Last Notified

Above we can observe our database represented in the Universal Relation Schema (URS). In this universal form, all data is consolidated into a single table as a conceptual starting point. This approach allows us to view the database holistically, which serves as the foundation for analyzing and structuring data. Using the URS, we can systematically work through the normalization process of First, Second and Third Normal Form, ensuring the database achieves an optimal design that minimizes redundancy and maintains data integrity

First Normal Form, Second Normal Form & Third Normal Form (1,2,3NF)

Looking at the definitions of each normal form below we can then assess our relational design

A table is in first normal form if it contains only atomic (indivisible) values, and each column contains values of a single type. Repeating groups and arrays are not allowed.(Elmasri & Navathe, 2015,: 477) Second Normal Form is a relation where all non primary key attributes are fully functionally dependent on the primary key while also surpassing First Normal Form.' (Connelly & Begg, 2005: 407). Third Normal Form is a relation in which no non-candidate-key attribute is transitively dependent on any candidate while also satisfying the requirements of First and Second Normal Form.' (Connelly & Begg, 2005: 411). Taking these definitions into account we can revise our table as below.

LOANS	(**Loans ID**, Member ID, Resource ID, Loan Date, Date Returned)
MEMBERS	(**Member ID,** Member Name, Member Email, Loan Limit)
BOOKS	**(Book ID,** ISBN, Title, Author, Genre, Year, Class ID, B. Loan Length, Book Quantity)
EBOOKS	(**eBook ID**, ISBN, Title, Author, Genre, Year, Eb. Loan Length, eBook Quantity)
EQUIPMENT	(**Equipment ID**, Type, Model, Make, E Loan Length, Equip Quantity, Loan Type)
RESOURCES	(**Resource ID**, Book ID, eBook ID, Equipment ID)
RESERVATIONS	(**Reservation ID**, Resource ID, Member ID, Notifications, Date Last Notified)
FINES	(**Fine ID**, Member ID, Amount)
CLASS	(**Class ID**, Floor, Shelf)

LOANS is both second and third normal forms. All attributes, including *Member ID*, *Resource ID, Loan Date* and *Date Returned*, are fully dependent on the primary key, *Loan ID*. There are no partial or transitive dependencies, ensuring compliance with both forms.

MEMBERS satisfies second and third normal forms. Each non-key attribute, such as *Name, Email* and *Loan Limit*, is fully dependent on the primary key, *Member ID* with no partial or transitive dependencies.

BOOKS is in second and third normal forms. All non-primary key attributes, including *ISBN, Title, Author, Genre, Year, Class ID, Loan Length*, and *B Quantity*, are fully dependent on the primary key, *Book ID*, with no partial or transitive dependencies.

EBOOKS meets second and third normal forms. Non-primary key attributes such as *ISBN, Title, Author, Genre, Year, Loan Length* and *eB Quantity* are fully dependent on the primary key, *eBook ID*, with no partial or transitive dependencies.

EQUIPMENT also satisfies second and third normal forms. Non-primary key attributes, including *Type, Model, Make, Loan Length, E Quantity* and *Loan Type*, are fully dependent on the primary key, *Equipment ID*, with no partial or transitive dependencies.

RESOURCES is compliant with second and third normal forms. Non-primary key attributes such as *Book ID, eBook ID, and Equipment ID* are fully dependent on the primary key, *Resource ID*, with no partial or transitive dependencies.

RESERVATIONS meets second and third normal forms. All non-primary key attributes: *Resource ID, Member ID, Notifications* and *Date Last Notified,* are fully dependent on the primary key, *Reservation ID*, with no partial or transitive dependencies.

FINES is in second and third normal forms. The non-primary key attribute, *Amount*, is fully dependent on the primary key, *Fine ID*, with no partial or transitive dependencies.

Finally, **CLASS** also satisfies both second and third normal forms. Non-key attributes, such as *Floor* and *Shelf*, are fully dependent on the primary key, *Class ID*, with no partial or transitive dependencies.

Bibliography

Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson Education. p. 477.

Connelly, C., & Begg, C. (2005). *Database Systems: A Practical Approach to Design, Implementation, and Management* (4th ed.). Pearson Education. p. 407.

Connelly, C., & Begg, C. (2005). *Database Systems: A Practical Approach to Design, Implementation, and Management* (4th ed.). Pearson Education. p. 411.

Adrian Bunn 241072051

James Kelly 240889087

Ruben Elbrond-Palmer 240946782

Database Implementation (Part 2)

```
CREATE TABLE MEMBERS (
   MEMBER_ID VARCHAR2(20) PRIMARY KEY,
   MEMBER_NAME VARCHAR2(100),
   MEMBER_EMAIL VARCHAR2(100),
   LOAN_LIMIT NUMBER);
CREATE TABLE BOOKS (
   BOOK_ID VARCHAR2(20) PRIMARY KEY,
   ISBN VARCHAR2(20),
   TITLE VARCHAR2(100),
   AUTHOR VARCHAR2(100),
   GENRE VARCHAR2(50),
   YEAR NUMBER,
   CLASS_ID VARCHAR2(10),
   B_LOAN_LENGTH VARCHAR2(20),
   BOOK_QUANTITY NUMBER);
CREATE TABLE EBOOKS (
   EBOOK_ID VARCHAR2(20) PRIMARY KEY,
   ISBN VARCHAR2(20),
   TITLE VARCHAR2(100),
   AUTHOR VARCHAR2(100),
   GENRE VARCHAR2(50),
   YEAR NUMBER,
   EB_LOAN_LENGTH VARCHAR2(20),
   EBOOK_QUANTITY NUMBER);
CREATE TABLE EQUIPMENT (
   EQUIPMENT_ID VARCHAR2(20) PRIMARY KEY,
   TYPE VARCHAR2(50),
   MODEL VARCHAR2(50),
   MAKE VARCHAR2(50),
   E_LOAN_LENGTH VARCHAR2(20),
   EQUIP_QUANTITY NUMBER,
   LOAN_TYPE VARCHAR2(50));
CREATE TABLE RESERVATIONS (
   RESERVATION_ID VARCHAR2(20) PRIMARY KEY,
   RESOURCE_ID VARCHAR2(20),
   MEMBER_ID VARCHAR2(20),
   NOTIFICATIONS NUMBER,
   DATE_LAST_NOTIFIED DATE);
CREATE TABLE RESOURCES (
   RESOURCE_ID VARCHAR2(20) PRIMARY KEY,
   BOOK_ID VARCHAR2(20),
   EBOOK_ID VARCHAR2(20),
   EQUIPMENT_ID VARCHAR2(20));
CREATE TABLE CLASS (
   CLASS_ID VARCHAR2(10) PRIMARY KEY,
   FLOOR NUMBER,
   SHELF VARCHAR2(10));
CREATE TABLE LOANS (
   LOANS_ID VARCHAR2(20) PRIMARY KEY,
   MEMBER_ID VARCHAR2(20),
   RESOURCE_ID VARCHAR2(20),
   LOAN_DATE DATE,
   DATE_RETURNED DATE);
CREATE TABLE FINES (
   FINE_ID VARCHAR2(20) PRIMARY KEY,
   MEMBER_ID VARCHAR2(20),
   AMOUNT NUMBER);
```

ALTER TABLES TO ADD FOREIGN KEY CONTSRAINT

```
ALTER TABLE RESOURCES
ADD CONSTRAINT fk_book_id FOREIGN KEY (BOOK_ID) REFERENCES BOOKS(BOOK_ID);
ALTER TABLE RESOURCES
ADD CONSTRAINT fk_ebook_id FOREIGN KEY (EBOOK_ID) REFERENCES EBOOKS(EBOOK_ID);
ALTER TABLE RESOURCES
ADD CONSTRAINT fk_equipment_id FOREIGN KEY (EQUIPMENT_ID) REFERENCES EQUIPMENT(EQUIPMENT_ID);
ALTER TABLE BOOKS
ADD CONSTRAINT fk_class_id FOREIGN KEY (CLASS_ID) REFERENCES CLASS(CLASS_ID);
ALTER TABLE RESERVATIONS
ADD CONSTRAINT fk_member_id FOREIGN KEY (MEMBER_ID) REFERENCES MEMBERS(MEMBER_ID);
ALTER TABLE RESERVATIONS
ADD CONSTRAINT fk_resource_id FOREIGN KEY (RESOURCE_ID) REFERENCES RESOURCES(RESOURCE_ID);
```

```
ALTER TABLE LOANS
ADD CONSTRAINT fk_loan_member_id FOREIGN KEY (MEMBER_ID) REFERENCES MEMBERS(MEMBER_ID);
ALTER TABLE LOANS
ADD CONSTRAINT fk_loan_resource_id FOREIGN KEY (RESOURCE_ID) REFERENCES RESOURCES(RESOURCE_ID);
ALTER TABLE FINES ADD CONSTRAINT fk_fines_member_id FOREIGN KEY (MEMBER_ID) REFERENCES MEMBERS(MEMBER_ID);
```

DATA POPULATION

```
INSERT INTO CLASS (CLASS_ID, FLOOR, SHELF) VALUES ('TEC', 1, 'A1');
INSERT INTO CLASS (CLASS_ID, FLOOR, SHELF) VALUES ('PRO', 1, 'A2');
INSERT INTO CLASS (CLASS_ID, FLOOR, SHELF) VALUES ('MATH', 1, 'A3');
INSERT INTO CLASS (CLASS_ID, FLOOR, SHELF) VALUES ('FICT', 2, 'B1');
INSERT INTO CLASS (CLASS_ID, FLOOR, SHELF) VALUES ('CLASS', 2, 'B2');
INSERT INTO CLASS (CLASS_ID, FLOOR, SHELF) VALUES ('LIT', 2, 'B3');

INSERT INTO BOOKS VALUES (110, '978-0123456', 'Deep Learning', 'Emma White', 'Tech', 2021, 'TEC', 21, 5);
INSERT INTO BOOKS VALUES (111, '978-2234567', 'Introduction to AI', 'Steve Rogers', 'Tech', 2020, 'TEC', 21, 7);
INSERT INTO BOOKS VALUES (112, '978-3345678', 'Machine Learning', 'Tony Stark', 'Programming', 2022, 'PRO', 21, 9);
INSERT INTO BOOKS VALUES (113, '978-4456789', 'Blockchain Basics', 'Natasha Romanoff', 'Tech', 2021, 'TEC', 21, 5);
INSERT INTO BOOKS VALUES (114, '978-5567890', 'Data Analytics', 'Bruce Banner', 'Tech', 2023, 'TEC', 21, 6);
INSERT INTO BOOKS VALUES (115, '978-6678901', 'Web Scraping', 'Wanda Maximoff', 'Tech', 2022, 'TEC', 21, 4);
INSERT INTO BOOKS VALUES (116, '978-7789012', 'Introduction to R', 'Clint Barton', 'Math', 2021, 'MATH', 21, 7);
INSERT INTO BOOKS VALUES (117, '978-8890123', 'Quantum Computing', 'Peter Parker', 'Tech', 2020, 'TEC', 21, 6);
INSERT INTO BOOKS VALUES (118, '978-9901234', 'Digital Marketing', 'Steve Rogers', 'Tech', 2022, 'TEC', 21, 8);
INSERT INTO BOOKS VALUES (119, '978-1012345', 'Cybersecurity Essentials', 'Tony Stark', 'Tech', 2021, 'TEC', 21, 10);
INSERT INTO BOOKS VALUES (120, '978-1113456', 'Cloud Security', 'Natasha Romanoff', 'Tech', 2023, 'TEC', 21, 4);
INSERT INTO BOOKS VALUES (121, '978-1237889', 'Pride and Prejudice', 'Jane Austen', 'Literature', 1813, 'CLASS', 21, 5);
INSERT INTO BOOKS VALUES (122, '978-2348990', '1984', 'George Orwell', 'Fiction', 1949, 'FICT', 21, 4);
INSERT INTO BOOKS VALUES (123, '978-3459101', 'Moby Dick', 'Herman Melville', 'Classics', 1851, 'CLASS', 21, 6);
INSERT INTO BOOKS VALUES (124, '978-4569212', 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 1925, 'FICT', 21, 7);

INSERT INTO EBOOKS VALUES ('E202', '978-0446573080', 'The Girl with the Louding Voice', 'Abi Daré', 'Fiction', 2020, '21 Days', 6);
INSERT INTO EBOOKS VALUES ('E203', '978-0385543132', 'The Four Winds', 'Kristin Hannah', 'Historical', 2021, '21 Days', 4);
INSERT INTO EBOOKS VALUES ('E204', '978-0307743657', 'The Silent Patient', 'Alex Michaelides', 'Thriller', 2019, '21 Days', 5);
INSERT INTO EBOOKS VALUES ('E205', '978-0385540803', 'American Dirt', 'Jeanine Cummins', 'Fiction', 2020, '21 Days', 7);
INSERT INTO EBOOKS VALUES ('E206', '978-1443419038', 'The Tattooist of Auschwitz', 'Heather Morris', 'Historical', 2018, '21 Days', 6);
INSERT INTO EBOOKS VALUES ('E207', '978-0399181894', 'The Chain', 'Adrian McKinty', 'Thriller', 2019, '21 Days', 5);
INSERT INTO EBOOKS VALUES ('E208', '978-0143129401', 'The Nightingale', 'Kristin Hannah', 'Historical', 2015, '21 Days', 8);
INSERT INTO EBOOKS VALUES ('E209', '978-0062952125', 'The Henna Artist', 'Alka Joshi', 'Fiction', 2020, '21 Days', 6);
INSERT INTO EBOOKS VALUES ('E210', '978-0062572112', 'Big Little Lies', 'Liane Moriarty', 'Fiction', 2014, '21 Days', 5);
INSERT INTO EBOOKS VALUES ('E211', '978-1419717015', 'A Man Called Ove', 'Fredrik Backman', 'Fiction', 2012, '21 Days', 7);
INSERT INTO EBOOKS VALUES ('E212', '978-0374533551', 'The Goldfinch', 'Donna Tartt', 'Fiction', 2013, '21 Days', 6);
INSERT INTO EBOOKS VALUES ('E213', '978-0452295292', 'Water for Elephants', 'Sara Gruen', 'Historical', 2006, '21 Days', 5);
INSERT INTO EBOOKS VALUES ('E214', '978-1499855406', 'The Paper Palace', 'Miranda Cowley Heller', 'Fiction', 2021, '21 Days', 4);
INSERT INTO EBOOKS VALUES ('E215', '978-1250769086', 'The Seven Husbands of Evelyn Hugo', 'Taylor Jenkins Reid', 'Fiction', 2017, '21 Days', 6);

INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010001', 'eReader', 'Paperwhite', 'Amazon', '3 Days', 37, 'Off-site');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010002', 'eReader', 'ColorSoft', 'Amazon', '3 Days', 18, 'Off-site');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010003', 'eReader', 'Fire', 'Amazon', '3 Days', 33, 'Off-site');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010004', 'Laptop', 'ThinkPad', 'Lenovo', '0 Days', 25, 'On-site Only');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010005', 'Laptop', 'L540', 'Lenovo', '0 Days', 25, 'On-site Only');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010006', 'Tablet', 'iPad', 'Apple', '0 Days', 42, 'On-site Only');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010007', 'Tablet', 'iPad Pro', 'Apple', '0 Days', 21, 'On-site Only');
INSERT INTO EQUIPMENT VALUES ('EQUIP-202401010008', 'Tablet', 'iPad Mini', 'Apple', '0 Days', 18, 'On-site Only');

INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010001', 110, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010002', 111, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010003', 112, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010004', 113, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010005', 114, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010006', 115, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010007', 116, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010008', 117, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010009', 118, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010010', 119, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010011', 120, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010012', 121, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010013', 122, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010014', 123, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010015', 124, NULL, NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010017', NULL, 'E202', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010018', NULL, 'E203', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010019', NULL, 'E204', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010020', NULL, 'E205', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010021', NULL, 'E206', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010022', NULL, 'E207', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010023', NULL, 'E208', NULL);
```

```
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010024', NULL, 'E209', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010025', NULL, 'E210', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010026', NULL, 'E211', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010027', NULL, 'E212', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010028', NULL, 'E213', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010029', NULL, 'E214', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010030', NULL, 'E215', NULL);
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010031', NULL, NULL, 'EQUIP-202401010001');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010032', NULL, NULL, 'EQUIP-202401010002');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010033', NULL, NULL, 'EQUIP-202401010003');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010034', NULL, NULL, 'EQUIP-202401010004');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010035', NULL, NULL, 'EQUIP-202401010005');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010036', NULL, NULL, 'EQUIP-202401010006');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010037', NULL, NULL, 'EQUIP-202401010007');
INSERT INTO RESOURCES (RESOURCE_ID, BOOK_ID, EBOOK_ID, EQUIPMENT_ID) VALUES ('RESO-202401010038', NULL, NULL, 'EQUIP-202401010008');

INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGJD001120919', 'John Doe', 'john.doe150994@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-PGMS001010921', 'Mary Smith', 'm.smith040195@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGAP001010922', 'Alice Price', 'a.price080694@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-STKA001010920', 'Kevin Adams', 'k.adams250896@ne.edu', 10);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGRB001010918', 'Robert Brown', 'r.brown200494@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-PGEV001010921', 'Emma Vance', 'e.vance300998@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGOL001010920', 'Olivia Lee', 'o.lee100599@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-STNP001010923', 'Nathan Patel', 'n.patel170123@ne.edu', 10);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGJB001010917', 'James Black', 'j.black030789@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-PGKW001010918', 'Kate Williams', 'k.williams290695@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGDC001010922', 'Daniel Cooper', 'd.cooper120499@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-STSB001010920', 'Sarah Brown', 's.brown060687@ne.edu', 10);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-PGFR001010923', 'Fiona Roberts', 'f.roberts250394@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-UGTD001010922', 'Thomas Drake', 't.drake220998@ne.edu', 5);
INSERT INTO MEMBERS (MEMBER_ID, MEMBER_NAME, MEMBER_EMAIL, LOAN_LIMIT)
VALUES ('NECLIB-STCC001010920', 'Chloe Collins', 'c.collins080297@ne.edu', 10);

INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010001', 'NECLIB-UGJD001120919', 'RESO-202401010001', TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2024-01-05', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010002', 'NECLIB-PGMS001010921', 'RESO-202401010005', TO_DATE('2024-01-02', 'YYYY-MM-DD'), TO_DATE('2024-01-06', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010003', 'NECLIB-UGAP001010922', 'RESO-202401010008', TO_DATE('2024-01-03', 'YYYY-MM-DD'), TO_DATE('2024-01-07', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010004', 'NECLIB-STKA001010920', 'RESO-202401010003', TO_DATE('2024-01-04', 'YYYY-MM-DD'), TO_DATE('2024-01-08', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010005', 'NECLIB-UGRB001010918', 'RESO-202401010019', TO_DATE('2024-01-05', 'YYYY-MM-DD'), TO_DATE('2024-01-09', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010006', 'NECLIB-PGEV001010921', 'RESO-202401010014', TO_DATE('2024-01-06', 'YYYY-MM-DD'), TO_DATE('2024-01-10', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010007', 'NECLIB-UGOL001010920', 'RESO-202401010007', TO_DATE('2024-01-07', 'YYYY-MM-DD'), TO_DATE('2024-01-11', 'YYYY-MM-DD'));
INSERT INTO Loans (Loans_ID, Member_ID, Resource_ID, Loan_Date, Date_Returned)
VALUES ('LOAN-202401010008', 'NECLIB-STNP001010923', 'RESO-202401010022', TO_DATE('2024-01-08', 'YYYY-MM-DD'), TO_DATE('2024-01-12', 'YYYY-MM-DD'));

INSERT INTO RESERVATIONS VALUES ('RESERV-202401010101', 'RESO-202401010019', 'NECLIB-UGJD001120919', 1, TO_DATE('2024-11-01', 'YYYY-MM-DD'));
INSERT INTO RESERVATIONS VALUES ('RESERV-202401010102', 'RESO-202401010020', 'NECLIB-PGMS001010921', 1, TO_DATE('2024-11-03', 'YYYY-MM-DD'));
INSERT INTO RESERVATIONS VALUES ('RESERV-202401010103', 'RESO-202401010021', 'NECLIB-UGAP001010922', 2, TO_DATE('2024-11-05', 'YYYY-MM-DD'));
INSERT INTO RESERVATIONS VALUES ('RESERV-202401010104', 'RESO-202401010018', 'NECLIB-STKA001010920', 1, TO_DATE('2024-11-07', 'YYYY-MM-DD'));
INSERT INTO RESERVATIONS VALUES ('RESERV-202401010105', 'RESO-202401010022', 'NECLIB-UGRB001010918', 1, TO_DATE('2024-11-09', 'YYYY-MM-DD'));
INSERT INTO RESERVATIONS VALUES ('RESERV-202401010106', 'RESO-202401010015', 'NECLIB-PGEV001010921', 2, TO_DATE('2024-11-11', 'YYYY-MM-DD'));

INSERT INTO FINES VALUES ('FI-202401010039', 'NECLIB-UGJD001120919', 15);
INSERT INTO FINES VALUES ('FI-202401010074', 'NECLIB-PGMS001010921', 5);
INSERT INTO FINES VALUES ('FI-202401010014', 'NECLIB-UGAP001010922', 10);
INSERT INTO FINES VALUES ('FI-202401010045', 'NECLIB-STKA001010920', 0);
INSERT INTO FINES VALUES ('FI-202401010087', 'NECLIB-UGRB001010918', 20);
INSERT INTO FINES VALUES ('FI-202401010024', 'NECLIB-PGEV001010921', 25);
```

```
INSERT INTO FINES VALUES ('FI-202401010070', 'NECLIB-UGOL001010920', 30);
INSERT INTO FINES VALUES ('FI-202401010056', 'NECLIB-STNP001010923', 5);
INSERT INTO FINES VALUES ('FI-202401010018', 'NECLIB-UGJB001010917', 10);
INSERT INTO FINES VALUES ('FI-202401010093', 'NECLIB-PGKW001010918', 0);
INSERT INTO FINES VALUES ('FI-202401010067', 'NECLIB-UGDC001010922', 20);
INSERT INTO FINES VALUES ('FI-202401010078', 'NECLIB-STSB001010920', 10);
INSERT INTO FINES VALUES ('FI-202401010061', 'NECLIB-PGFR001010923', 15);
INSERT INTO FINES VALUES ('FI-202401010081', 'NECLIB-UGTD001010922', 5);
INSERT INTO FINES VALUES ('FI-202401010008', 'NECLIB-STCC001010920', 25);
```

View summary of member's loans:

```
CREATE VIEW MemberLoanSummary AS
SELECT
  M.Member_ID,
  M.Member_Name,
  COUNT(L.Loans_ID) AS Loans_Out
FROM
  MEMBERS M
LEFT JOIN
  LOANS L ON M.Member_ID = L.Member_ID
WHERE
  L.Date_Returned IS NULL
GROUP BY
  M.Member_ID, M.Member_Name;
```

```
select * from MemberLoanSummary
```

MEMBER_ID	MEMBER_NAME	LOANS_OUT
NECLIB-UGTD001010922	Thomas Drake	0
NECLIB-PGKW001010918	Kate Williams	0
NECLIB-UGJB001010917	James Black	0

View most popular resources by loan count:

```
CREATE VIEW PopularResources AS
SELECT
  R.Resource_ID,
  CASE
    WHEN R.Book_ID IS NOT NULL THEN 'Book'
    WHEN R.eBook_ID IS NOT NULL THEN 'eBook'
    WHEN R.Equipment_ID IS NOT NULL THEN 'Equipment'
  END AS Resource_Type,
  COUNT(L.Loans_ID) AS Loan_Count
FROM
  RESOURCES R
LEFT JOIN
  LOANS L ON R.Resource_ID = L.Resource_ID
GROUP BY
  R.Resource_ID, R.Book_ID, R.eBook_ID, R.Equipment_ID
ORDER BY
  Loan_Count DESC;
```

```
select * from PopularResources
```

RESOURCE_ID	RESOURCE_TYPE	LOAN_COUNT
RESO-202401010019	eBook	1
RESO-202401010005	Book	1
RESO-202401010003	Book	1
RESO-202401010014	Book	1

View suspended members with amount owed:

```
CREATE VIEW SuspendedMembers AS
SELECT
  F.Member_ID,
  M.Member_Name,
  F.Amount
FROM
  FINES F
JOIN
  MEMBERS M ON F.Member_ID = M.Member_ID
WHERE
  F.Amount >= 10;
```

```
select * from SuspendedMembers
```

MEMBER_ID	MEMBER_NAME	AMOUNT
NECLIB-UGJD001120919	John Doe	15
NECLIB-UGAP001010922	Alice Price	10
NECLIB-UGRB001010918	Robert Brown	20
NECLIB-PGEV001010921	Emma Vance	25

Create 4 simple SQL queries that feature the keywords SELECT and WHERE. Outputs are displayed on the right of the queries as screenshots:

1) List all equipment available for Loan

```
SELECT
   E.Type,
   E.Model,
   E.Equip_Quantity
FROM
   EQUIPMENT E
WHERE
   E.Equip_Quantity > 0;
```

TYPE	MODEL	EQUIP_QUANTITY
eReader	Paperwhite	37
eReader	ColorSoft	18
eReader	Fire	33
Laptop	ThinkPad	25

2) List all eBooks written by Kristin Hannah

```
SELECT Title, Author, Year
 FROM EBOOKS
 WHERE Author = 'Kristin Hannah';
```

TITLE	AUTHOR	YEAR
The Four Winds	Kristin Hannah	2021
The Nightingale	Kristin Hannah	2015

3) List all books where the genre is tech

```
SELECT Title, Author
 FROM BOOKS
 WHERE Genre = 'Tech';
```

TITLE	AUTHOR
Deep Learning	Emma White
Introduction to AI	Steve Rogers
Blockchain Basics	Natasha Romanoff
Data Analytics	Bruce Banner
Web Scraping	Wanda Maximoff
Quantum Computing	Peter Parker

4) retrieves the names of members and their loan limits where the loan limit exceeds 5

```
SELECT Member_Name, Loan_Limit

FROM MEMBERS WHERE Loan_Limit > 5;
```

MEMBER_NAME	LOAN_LIMIT
Kevin Adams	10
Nathan Patel	10
Sarah Brown	10
Chloe Collins	10

Create 4 intermediate SQL queries that feature the keywords JOIN:1) List of all books in the fiction category

```
SELECT
   B.Title,
   B.Author,
   C.Floor,
   C.Shelf
FROM
   BOOKS B
JOIN
   CLASS C ON B.Class_ID = C.Class_ID
WHERE
   C.Class_ID = 'FICT';
```

TITLE	AUTHOR	FLOOR	SHELF
1984	George Orwell	2	B1
The Great Gatsby	F. Scott Fitzgerald	2	B1

2) Fine amount above 5 pounds per member

```
SELECT M.Member_Name, F.Amount
FROM MEMBERS M
JOIN FINES F ON M.Member_ID = F.Member_ID
WHERE F.Amount > 5;
```

MEMBER_NAME	AMOUNT
John Doe	15
Alice Price	10
Robert Brown	20
Emma Vance	25
Olivia Lee	30
James Black	10
Daniel Cooper	20
Sarah Brown	10
Fiona Roberts	15
Chloe Collins	25

3) Retrieves the member name, resource ID, loan date, and return date for loans made after January 1, 2024.

```
SELECT
   M.Member_Name,
   R.Resource_ID,
   L.Loan_Date,
   L.Date_Returned
FROM
   LOANS L
JOIN MEMBERS M ON L.Member_ID = M.Member_ID
JOIN RESOURCES R ON L.Resource_ID = R.Resource_ID
WHERE   L.Loan_Date > '01-JAN-2024'4) Retrieve all overdue loans with member details
```

MEMBER_NAME	RESOURCE_ID	LOAN_DATE	DATE_RETURNED
Mary Smith	RESO-202401010005	02-JAN-24	06-JAN-24
Alice Price	RESO-202401010008	03-JAN-24	07-JAN-24
Kevin Adams	RESO-202401010003	04-JAN-24	08-JAN-24
Robert Brown	RESO-202401010019	05-JAN-24	09-JAN-24
Emma Vance	RESO-202401010014	06-JAN-24	10-JAN-24
Olivia Lee	RESO-202401010007	07-JAN-24	11-JAN-24
Nathan Patel	RESO-202401010022	08-JAN-24	12-JAN-24

```
SELECT
   L.Loans_ID,
   M.Member_Name, M.Member_Email
FROM
   LOANS L
JOIN
    MEMBERS M ON L.Member_ID = M.Member_ID
WHERE
SYSDATE - L.Loan_Date
```

LOANS_ID	MEMBER_NAME	MEMBER_EMAIL
LOAN-202401010001	John Doe	john.doe150994@ne.edu
LOAN-202401010002	Mary Smith	m.smith040195@ne.edu
LOAN-202401010004	Kevin Adams	k.adams250896@ne.edu
LOAN-202401010005	Robert Brown	r.brown200494@ne.edu
LOAN-202401010006	Emma Vance	e.vance300998@ne.edu
LOAN-202401010003	Olivia Lee	o.lee100599@ne.edu
LOAN-202401010007	Olivia Lee	o.lee100599@ne.edu
LOAN-202401010008	Nathan Patel	n.patel170123@ne.edu
LOAN-202401010009	James Black	j.black030789@ne.edu
LOAN-202401010010	Kate Williams	k.williams290695@ne.edu

Create 4 advanced queries that feature the keywords JOIN and GROUP BY.

1) Count of Loans Per Member

```
SELECT
   M.Member_Name,
   COUNT(L.Loans_ID) AS Total_Loans
FROM
   MEMBERS M
JOIN
   LOANS L ON M.Member_ID = L.Member_ID
GROUP BY
   M.Member_Name
ORDER BY
   Total_Loans DESC;
```

MEMBER_NAME	TOTAL_LOANS
Nathan Patel	2
Olivia Lee	2
Robert Brown	1
Emma Vance	1
Kate Williams	1
Mary Smith	1
John Doe	1
Kevin Adams	1
James Black	1

2) **Books Loaned by Class.**

```
                SELECT
   C.Class_ID,
   COUNT(L.Loans_ID) AS Loan_Count
FROM
   LOANS L
JOIN
   RESOURCES R ON L.Resource_ID = R.Resource_ID
JOIN
   BOOKS B ON R.Book_ID = B.Book_ID
JOIN
   CLASS C ON B.Class_ID = C.Class_ID
GROUP BY
   C.Class_ID;
```

CLASS_ID	LOAN_COUNT
FICT	1
TEC	3
CLASS	2
PRO	1
MATH	1

3) **Retrieves a list of members along with the total number of loans they have made, ordered by the highest number of loans**

```
SELECT m.member_id,   m.member_name AS member_name,
COUNT(l.loans_id) AS total_loans
FROM members m
JOIN loans l ON m.member_id = l.member_id
GROUP BY
m.member_id, m.member_name
ORDER BY total_loans DESC;
```

MEMBER_ID	MEMBER_NAME	TOTAL_LOANS
NECLIB-UGOL001010920	Olivia Lee	2
NECLIB-STNP001010923	Nathan Patel	2
NECLIB-STKA001010920	Kevin Adams	1
NECLIB-PGMS001010921	Mary Smith	1
NECLIB-UGJB001010917	James Black	1
NECLIB-PGEV001010921	Emma Vance	1
NECLIB-PGKW001010918	Kate Williams	1
NECLIB-UGJD001120919	John Doe	1

4) Calculates fines owed by each member by summing overdue penalties.

```
SELECT m.member_id, m.member_name AS member_name,
SUM(f.amount) AS total_fines
FROM members m
JOIN fines f ON m.member_id = f.member_id
GROUP BY m.member_id, m.member_name
HAVING SUM(f.amount) > 0
ORDER BY total_fines DESC;
```

MEMBER_ID	MEMBER_NAME	TOTAL_FINES
NECLIB-UGOL001010920	Olivia Lee	30
NECLIB-PGEV001010921	Emma Vance	25
NECLIB-STCC001010920	Chloe Collins	25
NECLIB-UGDC001010922	Daniel Cooper	20
NECLIB-UGRB001010918	Robert Brown	20
NECLIB-UGJD001120919	John Doe	15
NECLIB-PGFR001010923	Fiona Roberts	15
NECLIB-UGAP001010922	Alice Price	10
NECLIB-UGJB001010917	James Black	10
NECLIB-STSB001010920	Sarah Brown	10
NECLIB-UGTD001010922	Thomas Drake	5
NECLIB-STNP001010923	Nathan Patel	5
NECLIB-PGMS001010921	Mary Smith	5