Project Final (Database Initial Study / Database Design / Physical Design)

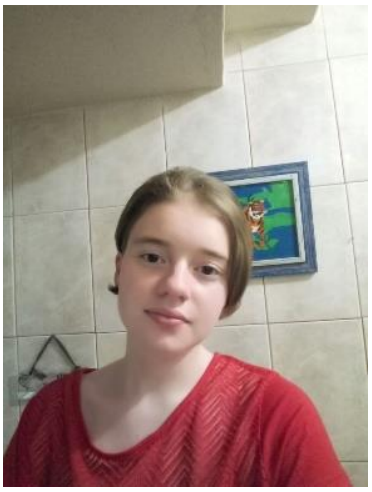
Table of Contents

Project Phase 1 – Database Initial Study

1. Introduction

i. Project Members

Group leader:



Name: Lizzon-Vanique De Villiers
Student Number: 40604012
Email: lvdv4j@gmail.com

Leon Mostert
Student Number: 20805330
Email: ernestmostert0202@gmail.com

Name: Henri Erasmus
Student Number: 35314834
Email: henrierasmus25@gmail.com

Name: Shaun Brits
Student Number: 40563189
Email: shaun@tegnon.co.za

ii. Background Information

Untold Stories is a privately owned library based in a small community in South Africa. The recently built library has librarians who serve the local community by lending out books to their members. A membership fee is charged which permits the library members to borrow books for four weeks at a

time. A penalty fee is charged for late returns as well as damaged or lost books. The library also allows members to reserve books that are currently being borrowed by someone else on a first come first serve basis.

Currently Untold Stories makes use of a manual filing system; storing all files on books, members, staff, and other general information in ring files stored within filing cabinets. However, this has proven to result in inefficient data retrieval and has also caused data anomalies and data inconsistency in the capturing process of this data. And due to the files being so easily accessible, there is a lack of data integrity.

2. Analyse company situation

i. Company Objectives

Untold Stories' mission is to rent out books quickly and easily to its members thus encouraging the community to become more interested in reading. To achieve this mission, Untold Stories must create a database to keep track of all the books that they are borrowing out, the status of these books, when the books are expected to be returned and the outstanding penalty fees charged to members for late/unreturned books. The system should also be able to track information regarding

the librarians who handled the returns or borrowing of a book, as well as information regarding the member borrowing the book. This will ensure that librarians are held responsible for the returns of the books they lent out and that they are only being paid when they work. It will also ensure that a member's book isn't registered as returned or outstanding more than once. An automated system to track all the information is integral to ensure accurate data is being captured and communicated to staff and clients.

ii. Company operations

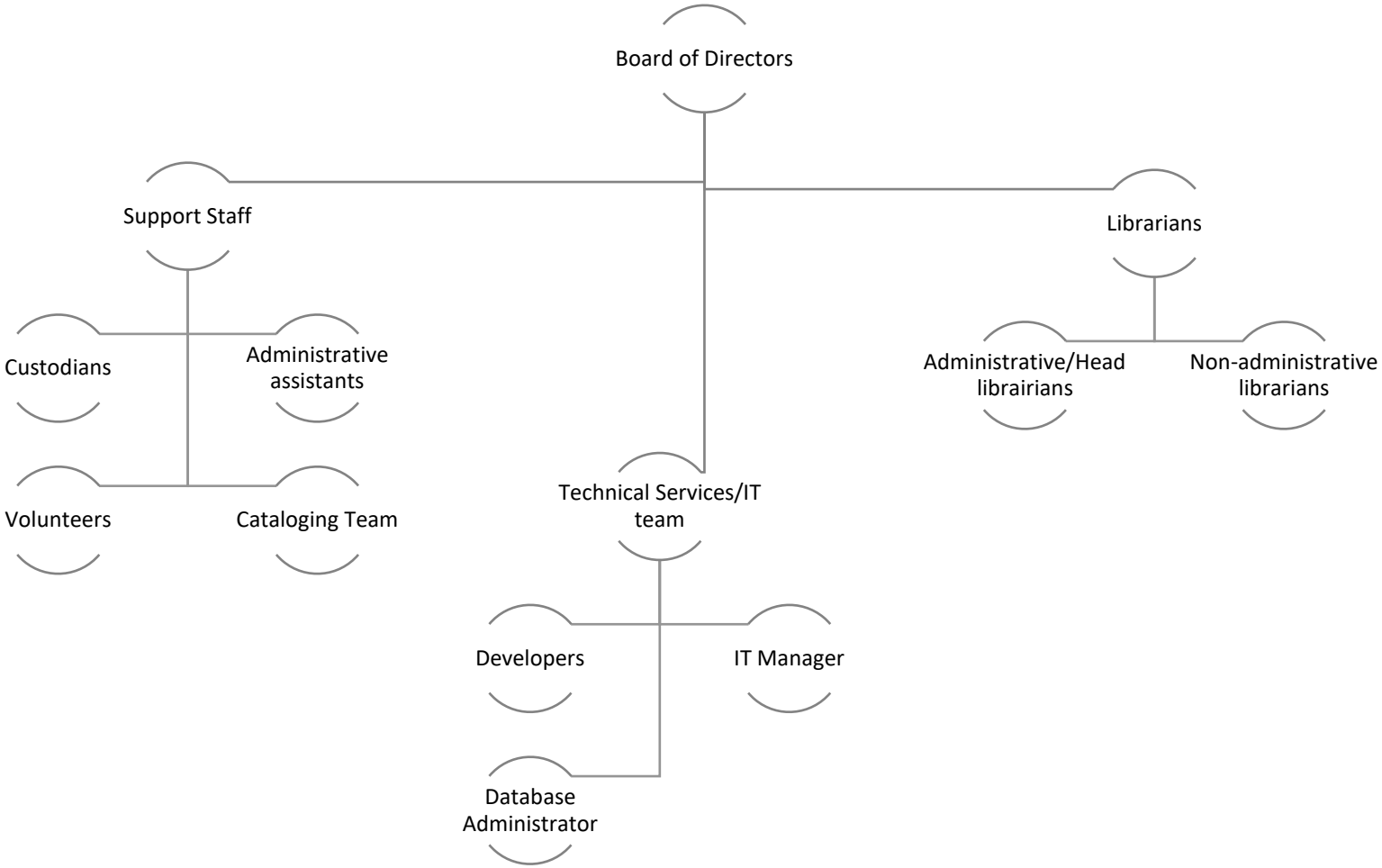
To implement a successful system, it is important to understand the library's operational flow:

When a new client becomes a member of the library, all personal information is captured and stored and a profile for the client is created.
All the current as well as newly hired librarian's details are stored as well, to track all changes made by the librarians.
Book information on all books is also stored on the system.
The library staff would be responsible for selecting, acquiring, and organizing books. This process may involve researching current trends in literature and popular demand from library members, evaluating the quality of the materials, and making decisions about how many copies to purchase.
When a client rents a book, the status/availability of the book changes. It is noted who is renting the book, the librarians that administered the transaction and a return date is generated.
If a client does not return the book on time, it is noted, and an additional fee is added to the client's account.
Librarians can check the status of books, generate reports on popular books and generate reports on outstanding or late books.

Business Rules:

1. Membership fee must be paid to borrow books from the library.
2. Members are allowed to borrow books for a maximum of four weeks at a time.
3. A penalty fee will be charged for late returns.
4. Members will be charged for lost or damaged books.
5. The library operates on a first come first serve basis for reserving books that are currently being borrowed by someone else.
6. The library maintains records of books, members, staff, and other general information.
7. All members must have a unique membership ID assigned to them upon registration.
8. Each book must have a unique book ID assigned to it upon entry into the inventory.
9. It is important to keep track of the reservation history of each book, including the date it was reserved and the member who reserved it.
10. Late fees must be calculated automatically by the system based on the return date of the borrowed book.
11. The database should have security measures in place to prevent unauthorized access to sensitive information and should provide a user-friendly interface for librarians.

iii. Company organizational structure



- Board of Directors: As a privately owned company, Untold Stories has a board of directors or owners who oversee the library's operations and finances.
- Support staff: support staff consists of administrative assistants or volunteers who help with tasks like shelving books, managing the library's social media accounts, or assisting with events.
- Librarians: The librarians are responsible for managing the day-to-day operations of the library, including lending out books, collecting fees, managing reservations, and maintaining the library's collection. They consist of both a head librarian and a non-administrative librarian. The head librarian will be able to access all member/library data in the database without restriction.
- Technical services/IT team: This team is responsible for implementing and maintaining a more efficient system for data storage and retrieval. The IT Manager is responsible for overseeing the IT team's activities and ensuring that the library's IT systems are functioning smoothly. The Database Administrator is responsible for managing the library's databases, ensuring data accuracy, and maintaining data security. And the developers are responsible for developing and customizing software applications to meet the library's specific needs.

3. Define problems and constraints

- Untold Stories currently uses a manual information system which has become inadequate due to an influx in members and stock. The following manual business processes have become time consuming and difficult to manage when being done:
    - Inventory (book) tracking
    - Member cash transactions
- Reporting is insufficient, e.g., there is no member information report, report of outstanding penalty fees, report on outstanding/popular books, etc.
- There is no search or sort function for book inventory, making it difficult for librarians to locate books.
- Members are continuing to borrow books regardless of outstanding penalty fees or the number of books they are currently in possession of.
- There is no effective way of tracking how many books a member is borrowing at any point in time.
- There is poor **communication** between librarians regarding which books have been returned and borrowed/loaned.
- The librarians often get files confused and/or lost, resulting in new files unnecessarily being created resulting in **data redundancy.**
- Physical space for files is limited thus making it difficult store a lot of files.
- Physical files are always accessible and available thus there is no proper **security** or access control with regards to who can view the data.
- Everyone has their own set of working records to use when they are on duty thus often resulting in different versions of the same data being recorded (**data anomalies**).
- The library struggles to determine the most popular genres and authors in demand among the customers.

- Manual transactions take several minutes to process, resulting in queues and long waiting times for members.

- To correct a member's record if an error was made, it takes a long-time as the librarians are forced to correct the error in every file where that member's information has been stored.
- The library may not have access to new and popular books, leading to long wait times for members hoping to borrow those titles.

- The costs to create the database may not exceed the amount set by Untold Stories' budget.

4. Database system specification

i. Define objectives to solve problems identified

The library must:

- Implement a system to improve business processes by automating status tracking of books to replace the manual system.
- Develop a reporting system that can generate reports on member information, outstanding penalty fees, outstanding/popular books, and other relevant data to facilitate decision making.
- Implement a search and sort function for the book inventory to improve the efficiency of locating books.
- Create a system that allows the librarians to track how many books a member is borrowing at any point in time.
- Improve business communication by providing a consistent graphical user interface which will allow librarians to update the status of borrowed and returned books.
- Store all records in a database which can be easily and quickly accessed by the system to be developed.
- Ensure that this database has access controls to determine which staff members can see which tables and to which extent they can be seen. This should also determine which staff members can edit the data and information and to what extent they can do so.
- Ensure that the system has data validation in place on input values to prevent errors.
- Develop a system that allows the librarians to easily correct errors in member records.
- Implement a faster and more efficient transaction processing system to reduce waiting times for members.
- Migrate all existing data to the new database system.
- Backup the database frequently to prevent a loss of data.
- Ensure that the payment for damages/lost/late books is received and establish a penalty fee system that prohibits members from borrowing more books or checking out books until they return the overdue books and pay the outstanding penalty fees.
- Ensure that it is in possession of the most up-to date and demanded books.
- Ensure that the data captured in the database should allow for the extrapolation of the most popular/in demand member trends and book preferences.

ii. Information that the company requires from database

- Inventory/Book Information: The database should provide up-to-date information about the library's book inventory, including the title, author, genre, publisher, ISBN, date of publication, number of copies available, the number of copies currently on loan, the number of books damaged, and the number of books that are lost.
- Reporting Information: The database should provide information regarding the outstanding books per time period as well as the top ten most popular books per time period.
- Member information: The database should store the personal information of all library members, including their full name, address, phone number, email address, and membership status. Included in this information should be how many books a member currently has and the outstanding fees a member owes.
- Penalty fee information: The database should track all penalty fees owed by members for late, lost, or damaged books.
- Librarian information: The database should store the personal information of all librarians, including their full name, address, phone number, email address, and their administrative privileges/roles. This will also include their usernames and passwords needed for them to access the system.
- Author information: The database should contain information on the authors of the books in the library, this includes their names, date of birth/death, and the number/list of books currently available from that author within the library's collection.
- Genre information: The database should contain information on the genres within the library as well as the list/number of books within each category.
- Reservation information: This would allow librarians to track which books are currently reserved and for whom. It could also help them manage the loan period for the book. Additionally, tracking reservations in the database could help librarians to identify popular books that are in high demand and potentially purchase additional copies to meet the needs of their members.

iii. Scope

The system must include functionality for the following:

- Maintenance of books.
- Maintenance of members.
- Maintenance of librarians.
- Maintenance of authors.
- Maintenance of reservations
- Borrowing of books by members.
- Returning of books by members.
- Receiving payments.
- Extensive reporting, including the top 10 books at any given time, as well as the current books overdue.
- Access control with different levels of access.

iv. Boundaries

- The database must be implemented and designed within a budget of R150 000.00.
- Due to the budgetary constraint, it will be necessary to make use of open-source software and design a system that will be able to run on a variety of hardware.
- There is currently no digital inventory management system in place; this will add overhead cost to load the current inventory onto the system.
- The system will need to be easy to use and understand as the current staff at the library are not technically proficient and do not have a lot of experience on these types of systems.
- Database implementation should be limited to 2 months.
- All the librarians' computers should be operating on a 64bit Windows 10 operating system or higher as is the requirements of the software to be used.
- Governmental laws and regulations might prevent the ownership of certain book titles/material thus they will not be able to be offered within the library.
- There is only a limited number of people who will be able to develop the database due to the budgetary constraints.

Project Phase 2 – Database Design

1. Conceptual design

i. Business rules

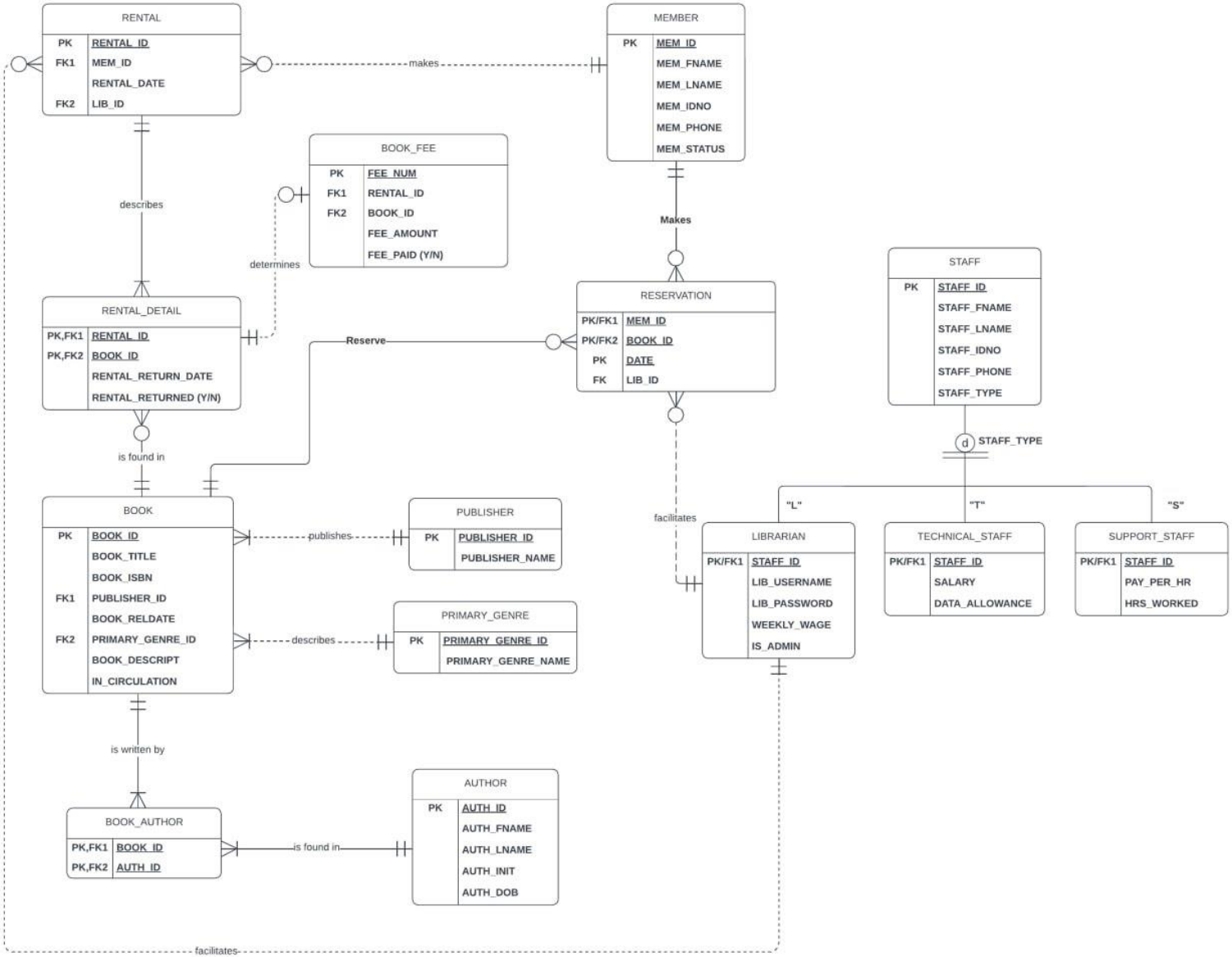
The following business rules should be considered when designing the database:

- When a member registers with the library for the first time the following personal information will be recorded: first name, last name, ID number, phone number and a unique member ID will be assigned to them. Their status will also be set to active.
- When a new staff member is hired, the following personal information will be recorded: first name, last name, ID number, phone number and a unique staff ID will be assigned to them.
- When a book is bought by the library, the relevant information on the book is recorded. This includes the book's ISBN number, title, publisher, author, release date, a short description of the book, and the library keeps track of whether a book might still be in circulation.
- A book has at least one author but can have multiple authors. Each author has written at least one book but can write multiple books.
- The following information on the author must also be recorded: first name, last name, initials, and date of birth. A unique author ID will also be assigned to them.
- A book can have many authors and an author can write many books.
- Each book has a publisher, and a publisher can publish many books.
- Each book has a primary genre, and a primary genre can describe many books.
- The staff consists of support staff, technical staff, and librarians.
    - Librarians can either have an administrative or non-administrative role determining their level of access to the system. As librarians will be using the system, they'll need a username and password.
    - Librarians are paid a weekly wage.
    - Technical staff have a fixed salary and are provided with a data allowance.
    - Support staff are paid on an hourly basis.
- When a member rents a book, the date on which the transaction occurred is recorded. For historical purposes it is also important to note the librarian who facilitated this transaction. Any librarian can facilitate many of these transactions, while some librarians have not facilitated any yet. A member can make many transactions, but they are not required to make any transactions to maintain their membership.
- Many books can be rented out in a single rental transaction and many rental transactions can be made for a book.
- The library operates on a first come first serve basis for reserving books that are currently being borrowed by someone else. A member can choose to reserve a book that is currently checked out by another member, and they can make multiple reservations if they desire. However, making a reservation is not mandatory and a member can choose not to make any reservations at all. If a book is popular and thus unavailable, it might be reserved multiple times. Not all books have been reserved.

- For recording purposes, it is important to take note of the librarian who facilitated the reservation request made by a member, as well as the date on which the reservation was made. A librarian can facilitate many of these reservation requests, some librarians might not have done so yet. A reservation is always recorded by one librarian.

- Members can borrow a book for a period of 4 weeks, after which a penalty fee will be incurred. Members will also be charged for books that have been damaged or lost.

ii. ER Diagram



iii. Notes on the ERD

Composite keys used in:

- BOOK_AUTHOR (BOOK_ID and AUTH_ID form the primary key)
- RENTAL_DETAIL (RENTAL_ID and BOOK_ID form the primary key)
- RESERVATION (MEM_ID and BOOK_ID and DATE form the primary key)

Weak entities:

- BOOK_AUTHOR
- RENTAL_DETAIL
- RESERVATION
- LIBRARIAN
- TECHNICAL_STAFF
- SUPPORT_STAFF

Bridge entities:

- RENTAL_DETAIL
- BOOK_AUTHOR
- RESERVATION

Strong relationships:

- STAFF and all its subtypes.
- MEMBER and RESERVATION
- BOOK and RESERVATION
- RENTAL and RENTAL_DETAIL
- RENTAIL_DETAIL and BOOK
- BOOK and BOOK_AUTHOR
- BOOK_AUTHOR and AUTHOR

Weak Relationships:

- All other relationships not mentioned above.

Mandatory relationships:

- A BOOK must have a PUBLISHER.
- A PUBLISHER must have a BOOK.
- A BOOK must have a PRIMARY_GENRE.
- A PRIMARY_GENRE must have a BOOK.
- A BOOK must have a BOOK_AUTHOR.
- A BOOK_AUTHOR must have a BOOK.
- An AUTHOR must have a BOOK_AUTHOR.
- A BOOK_AUTHOR must have an AUTHOR.
- A RENTAL must have a RENTAIL_DETAIL.
- A RENTAL_DETAIL must have a RENTAL.
- A RENTAL must have MEMBER.
- A RENTAL must have a LIBRARIAN.
- A RENTAL_DETAIL must have a BOOK.

- A BOOK_FEE must have a RENTAL_DETAIL.
- A RESERVATION must have a BOOK.
- A RESERVATION must have a MEMBER.
- A RESERVATION must have a LIBRARIAN.
- STAFF must be one of the following: a LIBRARIAN, TECHNICAL_STAFF or SUPPORT_STAFF.

Super entities:

- STAFF

Sub-type entities:

- LIBRARIAN
- TECHNICAL_STAFF
- SUPPORT_STAFF

2. Logical Design

i. Logical model

a. Strong Entities

- RENTAL(**RENTAL_ID**(PK), *MEM_ID*(FK1), RENTAL_DATE, *LIB_ID*(FK2))
    - PRIMARY KEY: RENTAL_ID
    - FOREIGN KEY: MEM_ID REFERENCES MEMBER
    - FOREIGN KEY: LIB_ID REFERENCES LIBRARIAN

- MEMBER(**MEM_ID**(PK), MEM_FNAME, MEM_LNAME, MEM_IDNO, MEM_PHONE, MEM_STATUS)
    - PRIMARY KEY: MEM_ID

- BOOK(**BOOK_ID**(PK), BOOK_TITLE, BOOK_ISBN, *PUBLISHER_ID*(FK1), BOOK_RELDATE, *PRIMARY_GENRE_ID*(FK2), BOOK_DESCRIPT, IN_CIRCULATION)
    - PRIMARY KEY: BOOK_ID
    - FOREIGN KEY: PUBLISHER_ID REFERENCES PUBLISHER
    - FOREIGN KEY: PRIMARY_GENRE_ID REFERENCES PRIMARY_GENRE

- PUBLISHER(**PUBLISHER_ID**(PK), PUBLISHER_NAME)
    - PRIMARY KEY: PUBLISHER_ID

- PRIMARY_GENRE(**PRIMARY_GENRE_ID**(PK), PRIMARY_GENRE_NAME)
    - PRIMARY KEY: PRIMARY_GENRE_ID

- BOOK_FEE(**FEE_NUM**(PK), *RENTAL_ID*(FK1), *BOOK_ID*(FK2), FEE_AMOUNT, FEE_PAID)
    - PRIMARY KEY: FEE_NUM
    - FOREIGN KEY: RENTAL_ID REFERENCES RENTAL
    - FOREIGN KEY: BOOK_ID REFERENCES BOOK

- AUTHOR(**AUTH_ID**(PK), AUTH_FNAME, AUTH_LNAME, AUTH_INIT, AUTH_DOB)
    - PRIMARY KEY: AUTH_ID

b. Supertype/Subtype Entities

- STAFF(**STAFF_ID**(PK), STAFF_FNAME, STAFF_LNAME, STAFF_IDNO, STAFF_PHONE, **STAFF_TYPE**)
    - PRIMARY KEY: STAFF_ID

- LIBRARIAN(***STAFF_ID***(PK)(FK1), LIB_USERNAME, LIB_PASSWORD, WEEKLY_WAGE, IS_ADMIN)
    - PRIMARY KEY: STAFF_ID
    - FOREIGN KEY: STAFF_ID REFERENCES STAFF

- TECHNICAL_STAFF(***STAFF_ID***(PK)(FK1), SALARY, DATA_ALLOWANCE)
    - PRIMARY KEY: STAFF_ID
    - FOREIGN KEY: STAFF_ID REFERENCES STAFF

- SUPPORT_STAFF(***STAFF_ID***(PK)(FK1), PAY_PER_HR, HRS_WORKED)
    - PRIMARY KEY: STAFF_ID
    - FOREIGN KEY: STAFF_ID REFERENCES STAFF

c. Weak Entities

- RENTAIL_DETAIL(***RENTAL_ID***(PK)(FK1), ***BOOK_ID***(PK)(FK2), RENTAL_RETURN_DATE, RENTAL_RETURNED)
    - PRIMARY KEY: RENTAL_ID, BOOK_ID
    - FOREIGN KEY: RENTAL_ID REFERENCES RENTAL
    - FOREIGN KEY: BOOK_ID REFERENCES BOOK

- RESERVATION(***MEM_ID***(PK)(FK1), ***BOOK_ID***(PK)(FK2), **DATE**(PK), *LIB_ID(FK3)*)
    - PRIMARY KEY: MEM_ID, BOOK_ID, DATE
    - FOREIGN KEY: MEM_ID REFERENCES MEMBER
    - FOREIGN KEY: BOOK_ID REFERENCES BOOK
    - FOREIGN KEY: LIB_ID REFERENCES LIBRARIAN

- BOOK_AUTHOR(***BOOK_ID***(PK)(FK1), ***AUTH_ID***(PK)(FK2))
    - PRIMARY KEY: BOOK_ID, AUTH_ID
    - FOREIGN KEY: BOOK_ID REFERENCES BOOK
    - FOREIGN KEY: AUTH_ID REFERENCES AUTHOR

ii. Validating Logical Model Integrity Constraints

- RENTAL
  - RENTAL_ID
    - Is a valid rental ID.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38
  - MEM_ID
    - Is a valid member ID and references a member.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38
  - RENTAL_DATE
    - Is a valid date.
    - Type: Date
    - Range: low value = 01/01/1800
    - High value = 31/12/9999
    - Display format: dd/mm/yyyy
  - LIB_ID
    - Is a valid librarian ID and references a librarian.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38

- RENTAIL_DETAIL
  - RENTAL_ID
    - Is a valid rental ID and references a rental.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38
  - BOOK_ID
    - Is a valid Book ID and references a book.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38
  - RENTAL_RETURN_DATE
    - Is a valid date.
    - Type: Date

        - Range: low value = 01/01/1800
        - High value = 31/12/9999
        - Display format: dd/mm/yyyy
    - RENTAL_RETURNED
        - Is a valid character code.
        - Type: character
        - Display format: X
        - Valid entries: Y, N (where Y represents yes and N represents no)
        - Length: 1

- MEMBER
    - MEM_ID
        - Is a valid member ID.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38

    - MEM_FNAME
        - Is a valid Member's first name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - MEM_LNAME
        - Is a valid member's last name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - MEM_IDNO
        - Is a valid Member's ID.
        - Type: character
        - Display format: XXXXXXXXXXXXXXX
        - Length = 13
    - MEM_PHONE
        - Is a valid Member's phone number.
        - Type: Numeric
        - Range: low value = 0000000000
        - High value = 9999999999
        - Display format = 9999999999
        - Length = 10
    - MEM_STATUS
        - Is a valid Member's status.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXXXX
        - Length: 10

- BOOK_FEE
    - FEE_NUM
        - Is a valid Fee Number ID
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - RENTAL_ID
        - Is a valid rental ID and references a rental.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - BOOK_ID
        - Is a valid Book ID and references a book.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - FEE_AMOUNT
        - Is a valid fee amount.
        - Type: Money
        - Range: low value = 000000.00
        - High value = 999999.99
        - Display format = 000000.00
    - FEE_PAID
        - Is a valid character code.
        - Type: character
        - Display format: X
        - Valid entries: Y, N (where Y represents yes and N represents no)
        - Length: 1

- RESERVATION
    - MEM_ID
        - Is a valid member ID and references a member.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - BOOK_ID
        - Is a valid Book ID and references a book.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999

        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - DATE
        - Is a valid date.
        - Type: Date
        - Range: low value = 01/01/1800
        - High value = 31/12/9999
        - Display format: dd/mm/yyyy
    - LIB_ID
        - Is a valid librarian ID and references a librarian.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38

- STAFF
    - STAFF_ID
        - Is a valid staff ID.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38

    - STAFF_FNAME
        - Is a valid staff member's first name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - STAFF_LNAME
        - Is a valid staff member's last name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - STAFF_IDNO
        - Is a valid staff member's ID.
        - Type: character
        - Display format: XXXXXXXXXXXXXXX
        - Length = 13
    - STAFF_PHONE
        - Is a valid staff member's phone number.
        - Type: Numeric
        - Range: low value = 0000000000
        - High value = 9999999999
        - Display format = 9999999999
        - Length = 10

- STAFF_TYPE
  - Is a valid Employee's Type Code
  - Type: character
  - Display format: X
  - Valid entries: L, S, T
  - Length: 1

- SUPPORT_STAFF
  - STAFF_ID
    - Is a valid staff ID and references a staff member.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38
  - PAY_PER_HR
    - Is a valid monetary amount.
    - Type: Money
    - Range: low value = 000000.00
    - High value = 999999.99
    - Display format = 000000.00
  - HRS_WORKED
    - Is a valid numeric amount.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38

- BOOK
  - BOOK_ID
    - Is a valid book ID.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38
  - BOOK_TITLE
    - Is a valid book's title.
    - Type: character
    - Display format: XXXXXXXXXXXXXXXX…
    - Length: 100
  - BOOK_ISBN
    - Is a valid book ISBN.
    - Type: character
    - Display format: XXXXXXXXXXXXX
    - Length: 13
  - PUBLISHER_ID

        - Is a valid publisher ID and references a publisher.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - BOOK_RELDATE
        - Is a valid date.
        - Type: Date
        - Range: low value = 01/01/1800
        - High value = 31/12/9999
        - Display format: dd/mm/yyyy
    - PRIMARY_GENRE_ID
        - Is a valid primary genre ID and references a primary genre.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - BOOK_DESCRIPTION
        - Is a valid description of the book.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 2000
    - IN_CIRCULATION
        - Is a valid character code.
        - Type: character
        - Display format: X
        - Valid entries: Y, N (where Y represents yes and N represents no)
        - Length: 1

- PUBLISHER
    - PUBLISHER_ID
        - Is a valid publisher ID.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - PUBLISHER_NAME
        - Is a valid publisher's name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100

- PRIMARY_GENRE
    - PRIMARY_GENRE_ID
        - Is a valid primary genre ID.

        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - PRIMARY_GENRE_NAME
        - Is a valid primary genre's name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100

- AUTHOR
    - AUTH_ID
        - Is a valid author ID.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - AUTH_FNAME
        - Is a valid author's first name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - AUTH_LNAME
        - Is a valid author's last name.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - AUTH_INIT
        - Is a valid author's initials.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 12
    - AUTH_DOB
        - Is a valid date.
        - Type: Date
        - Range: low value = 01/01/1800
        - High value = 31/12/9999
        - Display format: dd/mm/yyyy

- BOOK_AUTHOR
    - BOOK_ID
        - Is a valid book ID and references a book.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999

        - Length = 38

    - AUTH_ID
        - Is a valid author ID and references an author.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38

- LIBRARIAN
    - STAFF_ID
        - Is a valid staff ID and references a staff member.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999
        - Display format = 99999999999999999999999999999999999999
        - Length = 38
    - LIB_USERNAME
        - Is a valid username.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - LIB_PASSWORD
        - Is a valid password.
        - Type: character
        - Display format: XXXXXXXXXXXXXXXX…
        - Length: 100
    - WEEKLY_WAGE
        - Is a valid monetary amount.
        - Type: Money
        - Range: low value = 000000.00
        - High value = 999999.99
        - Display format = 000000.00
    - IS_ADMIN
        - Is a valid character code.
        - Type: character
        - Display format: X
        - Valid entries: Y, N (where Y represents yes and N represents no)
        - Length: 1

- TECHNICAL_STAFF
    - STAFF_ID
        - Is a valid staff ID and references a staff member.
        - Type: Numeric
        - Range: low value = 00000000000000000000000000000000000000
        - High value = 99999999999999999999999999999999999999

        - Display format = 999999999999999999999999999999999999
        - Length = 38

- SALARY
    - Is a valid salary amount.
    - Type: Money
    - Range: low value = 000000.00
    - High value = 999999.99
    - Display format = 000000.00
- DATA_ALLOWANCE
    - Is a valid data allowance amount.
    - Type: Numeric
    - Range: low value = 00000000000000000000000000000000000000
    - High value = 99999999999999999999999999999999999999
    - Display format = 99999999999999999999999999999999999999
    - Length = 38

Project Phase 3 – Physical Design

1. Database Objects

i. Tables

a. Removing all tables to allow for the creation of new tables:

To ensure the absence of duplicate tables, we employed SQL statements to drop/delete all pre-existing tables before creating our own. To ensure that the tables are removed without error we also cascade constraints. Our approach involves starting with the child entities and then proceeding to drop the parent entities. Following the removal of the tables, we also drop/delete the sequences, guaranteeing that our primary keys always begin at their designated starting values.

```
/*=============================================

DROP TABLES

=============================================*/

DROP TABLE MEMBER CASCADE CONSTRAINTS;

DROP TABLE PUBLISHER CASCADE CONSTRAINTS;

DROP TABLE PRIMARY_GENRE CASCADE CONSTRAINTS;

DROP TABLE AUTHOR CASCADE CONSTRAINTS;

DROP TABLE BOOK CASCADE CONSTRAINTS;

DROP TABLE STAFF CASCADE CONSTRAINTS;

DROP TABLE TECHNICAL_STAFF CASCADE CONSTRAINTS;

DROP TABLE SUPPORT_STAFF CASCADE CONSTRAINTS;

DROP TABLE LIBRARIAN CASCADE CONSTRAINTS;

DROP TABLE RENTAL CASCADE CONSTRAINTS;

DROP TABLE BOOK_FEE CASCADE CONSTRAINTS;

DROP TABLE RENTAL_DETAIL CASCADE CONSTRAINTS;

DROP TABLE RESERVATION CASCADE CONSTRAINTS;

DROP TABLE BOOK_AUTHOR CASCADE CONSTRAINTS;
```

```
/*=============================================
DROP SEQUENCES
=============================================*/
DROP SEQUENCE MEMBER_SEQ;
DROP SEQUENCE PUBLISHER_SEQ;
DROP SEQUENCE PRIMARY_GENRE_SEQ;
DROP SEQUENCE AUTHOR_SEQ;
DROP SEQUENCE BOOK_SEQ;
DROP SEQUENCE STAFF_SEQ;
DROP SEQUENCE TECHNICAL_STAFF_SEQ;
DROP SEQUENCE SUPPORT_STAFF_SEQ;
DROP SEQUENCE LIBRARIAN_SEQ;
DROP SEQUENCE RENTAL_SEQ;
DROP SEQUENCE BOOK_FEE_SEQ;
```

b. *Table Creation*

The following statements pertain to the creation of tables in our database. These statements involve assigning primary keys, establishing relationships with foreign keys, and selecting appropriate and efficient data types for each field.

MEMBER TABLE

The member table maintains records of all registered library members. It comprises the following fields: MEM_ID (a unique identifier for each member), MEM_FNAME (stores the member's first name), MEM_LNAME (stores the member's last name), MEM_IDNO (stores the member's identification number), MEM_PHONE (stores the member's phone number; this field is optional and can be left empty), and MEM_STATUS (indicates the member's active status).

```
CREATE TABLE MEMBER(
   MEM_ID INT PRIMARY KEY,
   MEM_FNAME VARCHAR2(100) NOT NULL,
   MEM_LNAME VARCHAR2(100) NOT NULL,
   MEM_IDNO VARCHAR2(13) NOT NULL CHECK (LENGTH(MEM_IDNO) = 13),
   MEM_PHONE VARCHAR2(15),
   MEM_STATUS VARCHAR2(10) NOT NULL CHECK (MEM_STATUS IN ('Active',
'Inactive'))
);
```

Additionally, we will generate a sequence named MEMBER_SEQ to automatically assign values to the MEM_ID field. By implementing this sequence, we reduce the potential for user errors during manual data entry. This guarantees the preservation of data integrity and ensures that each record is assigned a distinct and unique primary key.

```
CREATE SEQUENCE MEMBER_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

PUBLISHER TABLE

The publisher table is responsible for storing information about publishers in the library database. It includes the following fields: PUBLISHER_ID (an integer value serving as the primary key for each publisher record), and PUBLISHER_NAME (storing the name of the publisher). The PUBLISHER_ID field is mandatory and cannot be left empty, ensuring that each publisher record has a unique identifier. Similarly, the PUBLISHER_NAME field is also mandatory and must contain a valid name for the publisher. Similarly, to the above, we also create sequence for publishers.

```
CREATE TABLE PUBLISHER(
   PUBLISHER_ID INT PRIMARY KEY NOT NULL,
   PUBLISHER_NAME VARCHAR2(100) NOT NULL
);

CREATE SEQUENCE PUBLISHER_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

PRIMARY_GENRE TABLE

The PRIMARY_GENRE table is utilized for storing primary genre information within the library database. It comprises two fields: PRIMARY_GENRE_ID, an integer field serving as the primary key for each primary genre, and PRIMARY_GENRE_NAME, which holds the name of the primary genre. To facilitate the automatic generation of unique values for the PRIMARY_GENRE_ID field, a sequence named PRIMARY_GENRE_SEQ has been implemented.

```
CREATE TABLE PRIMARY_GENRE(
   PRIMARY_GENRE_ID INT PRIMARY KEY,
   PRIMARY_GENRE_NAME VARCHAR2(100)
);

CREATE SEQUENCE PRIMARY_GENRE_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

AUTHOR TABLE

The author table serves as a storage mechanism for author-related information in our library database. It consists of the following fields: AUTH_ID (an integer field serving as the primary key for each author record), AUTH_FNAME (storing the author's first name), AUTH_LNAME (storing the author's last name), AUTH_INIT (a character field representing the author's initials), and AUTH_DOB (a date field indicating the author's date of birth). To automatically generate unique values for the AUTH_ID field, a sequence named AUTHOR_SEQ has been created.

```
CREATE TABLE AUTHOR(
   AUTH_ID INT PRIMARY KEY,
   AUTH_FNAME VARCHAR2(100) NOT NULL,
   AUTH_LNAME VARCHAR2(100) NOT NULL,
   AUTH_INIT CHAR(12) NOT NULL,
   AUTH_DOB DATE
);

CREATE SEQUENCE AUTHOR_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

BOOK TABLE

The book table is responsible for storing information about the books in our library database. It comprises several fields: BOOK_ID (an integer field serving as the primary key for each book record), BOOK_TITLE (storing the title of the book), BOOK_ISBN (storing the ISBN number of the book), BOOK_DESCRIPT (a character field of maximum length 500, providing a description or summary of the book), and IN_CIRCULATION (a single character field with a constraint ensuring it can only contain 'Y' or 'N' to indicate if the book is in circulation or not). The book table also includes foreign key constraints on the PUBLISHER_ID and PRIMARY_GENRE_ID fields, referencing the PUBLISHER and PRIMARY_GENRE tables, respectively. These constraints ensure that the values stored in the book table for these fields correspond to valid publisher and primary genre IDs in their respective tables. To facilitate the automatic generation of unique values for the BOOK_ID field, a sequence named BOOK_SEQ has also been created.

```
CREATE TABLE BOOK(
   BOOK_ID INT PRIMARY KEY,
   BOOK_TITLE VARCHAR2(100) NOT NULL,
   BOOK_ISBN VARCHAR2(13) NOT NULL CHECK (LENGTH(BOOK_ISBN) = 13),
   PUBLISHER_ID INT,
   BOOK_RELDATE DATE,
   PRIMARY_GENRE_ID INT,
   BOOK_DESCRIPT VARCHAR2(2000),
   IN_CIRCULATION CHAR(1 BYTE) CHECK (IN_CIRCULATION IN ('Y', 'N')),
   FOREIGN KEY (PUBLISHER_ID) REFERENCES PUBLISHER(PUBLISHER_ID),
   FOREIGN KEY (PRIMARY_GENRE_ID) REFERENCES
PRIMARY_GENRE(PRIMARY_GENRE_ID)
);

CREATE SEQUENCE BOOK_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

STAFF TABLE

The staff table is responsible for storing information about staff members in the library database. It consists of the following fields: STAFF_ID (an integer field serving as the primary key for each staff member record), STAFF_FNAME (a character field of maximum length 100, storing the first name of the staff member), STAFF_LNAME (a character field of maximum length 100, storing the last name of the staff member), STAFF_IDNO (a character field with a length of 13, storing the identification number of the staff member), STAFF_PHONE (a character field of maximum length 10, storing the phone number of the staff member; this field is optional and can be left empty), and STAFF_TYPE (a single character field with a constraint ensuring it can only contain 'L', 'T', or 'S' to indicate the type of staff member, where L represents a librarian, T represents a member of technical staff and S represents a member of support staff. To facilitate the automatic generation of unique values for the STAFF_ID field, a sequence named STAFF_SEQ has been created.

```
CREATE TABLE STAFF(
   STAFF_ID INT PRIMARY KEY,
   STAFF_FNAME VARCHAR2(100) NOT NULL,
   STAFF_LNAME VARCHAR2(100) NOT NULL,
   STAFF_IDNO VARCHAR2(13) NOT NULL CHECK (LENGTH(STAFF_IDNO) = 13),
   STAFF_PHONE VARCHAR2(15),
   STAFF_TYPE CHAR(1 BYTE) NOT NULL CHECK (STAFF_TYPE IN ('L', 'T', 'S'))
);

CREATE SEQUENCE STAFF_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

STAFF SUBTYPE TABLES

The TECHNICAL_STAFF table represents a subtype of the staff table, specifically for technical staff members in the library. It includes the following fields: STAFF_ID (an integer field serving as the primary key for each technical staff member), SALARY (storing the salary of the technical staff member; it must be a positive value), and DATA_ALLOWANCE (indicating the data allowance of the technical staff member; it must be a positive value). The STAFF_ID field is a foreign key referencing the STAFF table's primary key.

The SUPPORT_STAFF table represents another subtype of the staff table, dedicated to support staff members. It contains the following fields: STAFF_ID (an integer field serving as the primary key for each support staff member), PAY_PER_HR (a floating-point number field representing the pay per hour for the support staff member; it must be a positive value), and HRS_WORKED (a floating-point number field indicating the number of hours worked by the support staff member; it must be a positive value). The STAFF_ID field is a foreign key referencing the STAFF table's primary key.

The LIBRARIAN table represents yet another subtype of the staff table, specifically for librarian staff members. It includes the following fields: STAFF_ID (an integer field serving as the primary key for each librarian staff member), LIB_USERNAME (storing the username of the librarian), LIB_PASSWORD (storing the password of the librarian), WEEKLY_WAGE (the weekly wage of the librarian; it must be a positive value), and IS_ADMIN (a single character field with a constraint allowing only 'Y' or 'N' values to indicate whether the librarian has administrative privileges). The STAFF_ID field is a foreign key referencing the STAFF table's primary key.

To ensure the uniqueness and automatic generation of values for the primary keys (STAFF_ID) of each subtype table, separate sequences have been created: TECHNICAL_STAFF_SEQ for TECHNICAL_STAFF, SUPPORT_STAFF_SEQ for SUPPORT_STAFF, and LIBRARIAN_SEQ for librarian. These sequences start at 1 and increment by 1 for each new record, with no maximum limit and no cycling back to the starting value.

```
CREATE TABLE TECHNICAL_STAFF(
   STAFF_ID INT PRIMARY KEY,
   SALARY FLOAT NOT NULL CHECK (SALARY > 0),
   DATA_ALLOWANCE FLOAT NOT NULL CHECK (DATA_ALLOWANCE > 0),
   FOREIGN KEY (STAFF_ID) REFERENCES STAFF(STAFF_ID)
);

CREATE SEQUENCE TECHNICAL_STAFF_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

```
CREATE TABLE SUPPORT_STAFF(
   STAFF_ID INT PRIMARY KEY,
   PAY_PER_HR FLOAT NOT NULL CHECK (PAY_PER_HR > 0),
   HRS_WORKED FLOAT NOT NULL CHECK (HRS_WORKED > 0),
   FOREIGN KEY (STAFF_ID) REFERENCES STAFF(STAFF_ID)
);

CREATE SEQUENCE SUPPORT_STAFF_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;

CREATE TABLE LIBRARIAN(
   STAFF_ID INT PRIMARY KEY,
   LIB_USERNAME VARCHAR2(100) NOT NULL,
   LIB_PASSWORD VARCHAR2(100) NOT NULL,
   WEEKLY_WAGE FLOAT NOT NULL CHECK (WEEKLY_WAGE > 0),
   IS_ADMIN CHAR(1) NOT NULL CHECK (IS_ADMIN IN ('Y', 'N')),
   FOREIGN KEY (STAFF_ID) REFERENCES STAFF(STAFF_ID)
);

CREATE SEQUENCE LIBRARIAN_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

RENTAL TABLE

The rental table is used to store information about book rentals within our library. It consists of the following fields: RENTAL_ID (an integer field serving as the primary key for each rental), MEM_ID (an integer field representing the member ID of the member who made the rental; it cannot be null), RENTAL_DATE (a date field indicating the date of the rental; it cannot be null), and LIB_ID (an integer field representing the librarian ID of the librarian who processed the rental; it cannot be null).

The MEM_ID field is a foreign key referencing the MEM_ID field in the MEMBER table, ensuring that the member making the rental exists in the system. The LIB_ID field is a foreign key referencing the STAFF_ID field in the LIBRARIAN table, ensuring that the librarian processing the rental exists in the system.

To ensure the uniqueness and automatic generation of values for the primary key (RENTAL_ID) of each rental, a sequence named RENTAL_SEQ has been created. The sequence starts at 1 and increments by 1 for each new record, with no maximum limit and no cycling back to the starting value.

```
CREATE TABLE RENTAL(
   RENTAL_ID INT PRIMARY KEY,
   MEM_ID INT NOT NULL,
   RENTAL_DATE DATE NOT NULL,
   LIB_ID INT NOT NULL,
   FOREIGN KEY (MEM_ID) REFERENCES MEMBER(MEM_ID),
   FOREIGN KEY (LIB_ID) REFERENCES LIBRARIAN(STAFF_ID)
);

CREATE SEQUENCE RENTAL_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

BOOK FEE TABLE

The BOOK_FEE table is used to track the fees associated with book rentals in the library. It includes the following fields: FEE_NUM (an integer field serving as the primary key for each fee), RENTAL_ID (an integer field indicating the rental ID to which the fee is associated; it cannot be null), BOOK_ID (an integer field representing the book ID for which the fee is applicable; it cannot be null), FEE_AMOUNT (a floating-point field storing the amount of the fee; it cannot be null and must be greater than 0), and FEE_PAID (a character field indicating whether the fee has been paid or not, with valid values 'Y' for paid and 'N' for not paid; it cannot be null).

The RENTAL_ID field is a foreign key referencing the RENTAL_ID field in the RENTAL table, ensuring that the fee is linked to a valid rental record. The BOOK_ID field is a foreign key referencing the BOOK_ID field in the BOOK table, ensuring that the fee is associated with a valid book.

To generate unique values for the primary key (FEE_NUM) of each fee automatically, a sequence named BOOK_FEE_SEQ has been created. The sequence starts at 1 and increments by 1 for each new record, with no maximum value and no cycling back to the starting value.

```
CREATE TABLE BOOK_FEE(
   FEE_NUM INT PRIMARY KEY,
   RENTAL_ID INT NOT NULL,
   BOOK_ID INT NOT NULL,
   FEE_AMOUNT FLOAT NOT NULL CHECK (FEE_AMOUNT > 0),
   FEE_PAID CHAR(1) NOT NULL CHECK (FEE_PAID IN ('Y', 'N')),
   FOREIGN KEY (RENTAL_ID) REFERENCES RENTAL(RENTAL_ID),
   FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID)
);

CREATE SEQUENCE BOOK_FEE_SEQ
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE;
```

RENTAL DETAIL TABLE

The RENTAL_DETAIL table serves as a bridge entity between the book and rental tables, capturing the details of book rentals and their corresponding return information. It includes the following fields: RENTAL_ID (an integer field referencing the rental ID from the rental table), BOOK_ID (an integer field referencing the book ID from the book table), RENTAL_RETURN_DATE (a date field indicating the date the rental was returned, which cannot be null), and RENTAL_RETURNED (a character field indicating whether the rental has been returned or not, with valid values 'Y' for returned and 'N' for not returned; it cannot be null).

The primary key of the RENTAL_DETAIL table is a composite key consisting of RENTAL_ID and BOOK_ID, ensuring uniqueness for each combination of rental and book.

The RENTAL_ID field is a foreign key referencing the RENTAL_ID field in the rental table, maintaining the integrity of the relationship between rentals and rental details. Similarly, the BOOK_ID field is a foreign key referencing the BOOK_ID field in the book table, ensuring that the book associated with the rental is valid.

```
CREATE TABLE RENTAL_DETAIL(
   RENTAL_ID INT,
   BOOK_ID INT,
   RENTAL_RETURN_DATE DATE NOT NULL,
   RENTAL_RETURNED CHAR(1) NOT NULL CHECK (RENTAL_RETURNED IN ('Y',
'N')),
   PRIMARY KEY (RENTAL_ID, BOOK_ID),
   FOREIGN KEY (RENTAL_ID) REFERENCES RENTAL(RENTAL_ID),
   FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID)
);
```

RESERVATION TABLE

The reservation table serves as a bridge entity between the book and member tables, representing the reservations made by library members for specific books. It includes the following fields: MEM_ID (an integer field representing the member ID), BOOK_ID (an integer field indicating the book ID), RESERVATION_DATE (a date field recording the date of the reservation), and LIB_ID (an integer field referring to the librarian ID associated with the reservation).

The combination of MEM_ID, BOOK_ID, and RESERVATION_DATE serves as the primary key for each reservation record, ensuring uniqueness. The MEM_ID field is a foreign key referencing the MEM_ID field in the member table, establishing the association with a valid member. Similarly, the BOOK_ID field is a foreign key referencing the BOOK_ID field in the book table, ensuring that the reservation corresponds to a valid book. Lastly, the LIB_ID field is a foreign key referencing the STAFF_ID field in the librarian table, indicating the librarian responsible for the reservation.

By using this bridge entity, the reservation table facilitates the association between members, books, and librarians in a many-to-many relationship.

```
CREATE TABLE RESERVATION(
   MEM_ID INT,
   BOOK_ID INT,
   RESERVATION_DATE DATE,
   LIB_ID INT,
   PRIMARY KEY (MEM_ID, BOOK_ID, RESERVATION_DATE),
   FOREIGN KEY (MEM_ID) REFERENCES MEMBER(MEM_ID),
   FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID),
   FOREIGN KEY (LIB_ID) REFERENCES LIBRARIAN(STAFF_ID)
);
```

BOOK_AUTHOR TABLE

The BOOK_AUTHOR table functions as a bridge entity between the author and book tables, establishing the relationship between authors and the books they have authored. It contains the following fields: BOOK_ID (an integer field representing the book ID) and AUTH_ID (an integer field representing the author ID).

The combination of BOOK_ID and AUTH_ID serves as the primary key for each record in the BOOK_AUTHOR table, ensuring uniqueness and preventing duplicate associations between books and authors. The BOOK_ID field is a foreign key referencing the BOOK_ID field in the book table, establishing the connection to a specific book. Similarly, the AUTH_ID field is a foreign key referencing the AUTH_ID field in the author table, indicating the corresponding author for the book.

By utilizing this bridge entity, the BOOK_AUTHOR table enables a many-to-many relationship between authors and books, allowing multiple authors to be associated with multiple books, and vice versa.

```
CREATE TABLE BOOK_AUTHOR(
   BOOK_ID INT NOT NULL,
   AUTH_ID INT NOT NULL,
   PRIMARY KEY (BOOK_ID, AUTH_ID),
   FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID),
   FOREIGN KEY (AUTH_ID) REFERENCES AUTHOR(AUTH_ID)
);
```

ii. Indexes

AUTHOR TABLE INDEX

```
CREATE INDEX IDX_AUTHOR_NAME ON AUTHOR(AUTH_FNAME, AUTH_LNAME);
```

We use this statement to create an index named " IDX_AUTHOR_NAME ". This index is created on the columns "AUTH_FNAME" and "AUTH_LNAME" within the "AUTHOR" table. By creating this index, the database system organizes the data in the "AUTHOR" table in a way that facilitates efficient searching and retrieval based on the values in the "AUTH_FNAME" and "AUTH_LNAME" columns. This index can improve the performance of queries that involve searching, sorting, or joining the "AUTHOR" table based on the first name and last name of authors.

BOOK TABLE INDEX

```
CREATE INDEX IDX_BOOK_NAME_AUTHOR ON BOOK(BOOK_TITLE, PUBLISHER_ID);
```

This creates an index named " IDX_BOOK_NAME_AUTHOR " on the "BOOK" table. This index is created on the columns "BOOK_TITLE" and "PUBLISHER_ID" within the "BOOK" table.

MEMBER TABLE INDEX

```
CREATE INDEX IDX_MEMBER_NAME ON MEMBER(MEM_FNAME, MEM_LNAME);
```

This creates an index named " IDX_MEMBER_NAME " on the "MEMBER" table. This index is created on the columns "MEM_FNAME" and "MEM_LNAME" within the "MEMBER" table.

iii. Data Loading

We populated our databases by utilizing SQL queries and hardcoded the data into the tables. Since we are populating all the fields in each table, we do not need to explicitly list the attributes following the INSERT INTO statement. Instead, we directly specified the corresponding values. Here are some examples of the queries we used. Please note that these examples include only a few instances of each query. In our actual code, we have multiple versions of these INSERT statements for each table, providing them with the necessary data for testing our queries.

<table>
<tr><th>TABLE NAME</th><th>EXAMPLES OF INSERT INTO QUERIES</th></tr>
<tr><td>MEMBER</td><td>INSERT INTO MEMBER
VALUES (MEMBER_SEQ.NEXTVAL, 'John', 'Doe', '1234567890123', '555-1234', 'Active');

INSERT INTO MEMBER
VALUES (MEMBER_SEQ.NEXTVAL, 'Jane', 'Smith', '0987654321012', '555-5678', 'Inactive');

INSERT INTO MEMBER
VALUES (MEMBER_SEQ.NEXTVAL, 'Michael', 'Johnson', '5432167890135', '555-2468', 'Active');

INSERT INTO MEMBER
VALUES (MEMBER_SEQ.NEXTVAL, 'Sarah', 'Williams', '6789054321846', '555-7890', 'Active');

INSERT INTO MEMBER
VALUES (MEMBER_SEQ.NEXTVAL, 'David', 'Brown', '6789054321642', '555-1357', 'Inactive');

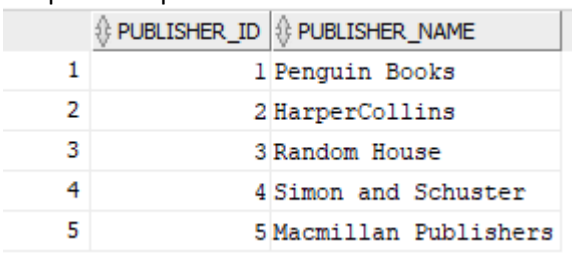
Output example:
<table>
<tr><th></th><th>MEM_ID</th><th>MEM_FNAME</th><th>MEM_LNAME</th><th>MEM_IDNO</th><th>MEM_PHONE</th><th>MEM_STATUS</th></tr>
<tr><td>1</td><td>1</td><td>John</td><td>Doe</td><td>1234567890123</td><td>555-1234</td><td>Active</td></tr>
<tr><td>2</td><td>2</td><td>Jane</td><td>Smith</td><td>0987654321012</td><td>555-5678</td><td>Inactive</td></tr>
<tr><td>3</td><td>3</td><td>Michael</td><td>Johnson</td><td>5432167890135</td><td>555-2468</td><td>Active</td></tr>
<tr><td>4</td><td>4</td><td>Sarah</td><td>Williams</td><td>6789054321846</td><td>555-7890</td><td>Active</td></tr>
<tr><td>5</td><td>5</td><td>David</td><td>Brown</td><td>6789054321642</td><td>555-1357</td><td>Inactive</td></tr>
</table>
</td></tr>
<tr><td>PUBLISHER</td><td>INSERT INTO PUBLISHER
VALUES (PUBLISHER_SEQ.NEXTVAL, 'Penguin Books');

INSERT INTO PUBLISHER
VALUES (PUBLISHER_SEQ.NEXTVAL, 'HarperCollins');

INSERT INTO PUBLISHER
VALUES (PUBLISHER_SEQ.NEXTVAL, 'Random House');

INSERT INTO PUBLISHER
VALUES (PUBLISHER_SEQ.NEXTVAL, 'Simon and Schuster');

INSERT INTO PUBLISHER
VALUES (PUBLISHER_SEQ.NEXTVAL, 'Macmillan Publishers');</td></tr>
</table>

<table>
<tr><td></td><td>Output example:
 
<table>
<tr><th></th><th>PUBLISHER_ID</th><th>PUBLISHER_NAME</th></tr>
<tr><td>1</td><td>1</td><td>Penguin Books</td></tr>
<tr><td>2</td><td>2</td><td>HarperCollins</td></tr>
<tr><td>3</td><td>3</td><td>Random House</td></tr>
<tr><td>4</td><td>4</td><td>Simon and Schuster</td></tr>
<tr><td>5</td><td>5</td><td>Macmillan Publishers</td></tr>
</table></td></tr>
<tr><td>PRIMARY_GENRE</td><td>INSERT INTO PRIMARY_GENRE
VALUES (PRIMARY_GENRE_SEQ.NEXTVAL, 'Fiction');

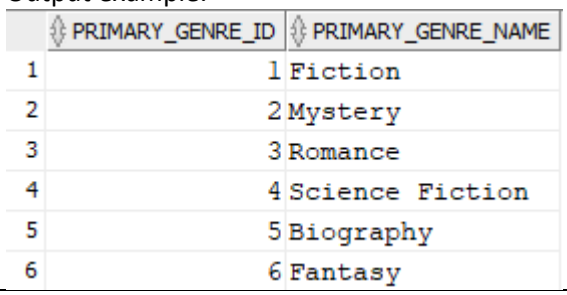
INSERT INTO PRIMARY_GENRE
VALUES (PRIMARY_GENRE_SEQ.NEXTVAL, 'Mystery');

INSERT INTO PRIMARY_GENRE
VALUES (PRIMARY_GENRE_SEQ.NEXTVAL, 'Romance');

INSERT INTO PRIMARY_GENRE
VALUES (PRIMARY_GENRE_SEQ.NEXTVAL, 'Science Fiction');

INSERT INTO PRIMARY_GENRE
VALUES (PRIMARY_GENRE_SEQ.NEXTVAL, 'Biography');

INSERT INTO PRIMARY_GENRE
VALUES (PRIMARY_GENRE_SEQ.NEXTVAL, 'Fantasy');

Output example:
 
<table>
<tr><th></th><th>PRIMARY_GENRE_ID</th><th>PRIMARY_GENRE_NAME</th></tr>
<tr><td>1</td><td>1</td><td>Fiction</td></tr>
<tr><td>2</td><td>2</td><td>Mystery</td></tr>
<tr><td>3</td><td>3</td><td>Romance</td></tr>
<tr><td>4</td><td>4</td><td>Science Fiction</td></tr>
<tr><td>5</td><td>5</td><td>Biography</td></tr>
<tr><td>6</td><td>6</td><td>Fantasy</td></tr>
</table></td></tr>
<tr><td>BOOK</td><td>INSERT INTO BOOK
VALUES (BOOK_SEQ.NEXTVAL, 'The Great Gatsby', '9780743273565', 1, TO_DATE('2022-01-15', 'YYYY-MM-DD'), 1, 'A classic novel by F. Scott Fitzgerald.', 'Y');

INSERT INTO BOOK
VALUES (BOOK_SEQ.NEXTVAL, 'Harry Potter and the Sorcerer''s Stone', '9780590353427', 2, TO_DATE('2001-10-01', 'YYYY-MM-DD'), 1, 'The first book in the Harry Potter series.', 'Y');

INSERT INTO BOOK
VALUES (BOOK_SEQ.NEXTVAL, 'Pride and Prejudice', '9780141439518', 3, TO_DATE('1813-01-28', 'YYYY-MM-DD'), 3, 'A classic romance novel by Jane Austen.', 'Y');

INSERT INTO BOOK
VALUES (BOOK_SEQ.NEXTVAL, 'Dune', '9780441172719', 4, TO_DATE('1965-06-01', 'YYYY-MM-DD'), 4, 'A science fiction novel by Frank Herbert.', 'Y');</td></tr>
</table>

	INSERT INTO BOOK VALUES (BOOK_SEQ.NEXTVAL, 'Steve Jobs', '9781451648539', 5, TO_DATE('2011-10-24', 'YYYY-MM-DD'), 5, 'A biography of Steve Jobs by Walter Isaacson.', 'Y'); INSERT INTO BOOK VALUES (BOOK_SEQ.NEXTVAL, 'The Eye of the World', '0312850093513', 2, TO_DATE('1990-01-15', 'YYYY-MM-DD'), 6, 'The first novel in a fantasy series by Robert Jordan.', 'Y'); INSERT INTO BOOK VALUES (BOOK_SEQ.NEXTVAL, 'Bloodstone', '1534856212579', 4, TO_DATE('1997-10-29', 'YYYY-MM-DD'), 6, 'The closing novel in a fantasy series by David Gemmell.', 'Y'); INSERT INTO BOOK VALUES (BOOK_SEQ.NEXTVAL, 'Destination: Void', '8456321856375', 5, TO_DATE('1966-06-15', 'YYYY-MM-DD'), 4, 'A science fiction novel by Frank Herbert.', 'Y'); INSERT INTO BOOK VALUES (BOOK_SEQ.NEXTVAL, 'Legend', '6485123578426', 3, TO_DATE('1984-03-27', 'YYYY-MM-DD'), 6, 'A fantasy novel by David Gemmell.', 'N'); INSERT INTO BOOK VALUES (BOOK_SEQ.NEXTVAL, 'The Great Hunt', '7512964822652', 1, TO_DATE('1990-11-15', 'YYYY-MM-DD'), 6, 'The second novel in a fantasy series by Robert Jordan.', 'N'); Output example (Please note that this is one table image that has been cropped into two)

	B...	BOOK_TITLE	BOOK_ISBN	P...	BOOK_RELDATE	P...	BOOK_DESCRIPT	IN_CI...
1	1	The Great Gatsby	9780743273565	1	15/JAN/22	1	A classic novel by F. Scott Fitzgerald.	Y
2	2	Harry Potter and...	9780590353427	2	01/OCT/01	1	The first book in the Harry Potter series.	Y
3	3	Pride and Prejudice	9780141439518	3	28/JAN/13	3	A classic romance novel by Jane Austen.	Y
4	4	Dune	9780441172719	4	01/JUN/65	4	A science fiction novel by Frank Herbert.	Y
5	5	Steve Jobs	9781451648539	5	24/OCT/11	5	A biography of Steve Jobs by Walter Isaacson.	Y
6	6	The Eye of the W...	0312850093513	2	15/JAN/90	6	The first novel in a fantasy series by Robe...	Y
7	7	Bloodstone	1534856212579	4	29/OCT/97	6	The closing novel in a fantasy series by Da...	Y
8	8	Destination: Void	8456321856375	5	15/JUN/66	4	A science fiction novel by Frank Herbert.	Y
9	9	Legend	6485123578426	3	27/MAR/84	6	A fantasy novel by David Gemmell.	N
10	10	The Great Hunt	7512964822652	1	15/NOV/90	6	The second novel in a fantasy series by Rob...	N

AUTHOR	INSERT INTO AUTHOR VALUES (AUTHOR_SEQ.NEXTVAL, 'J.K.', 'Rowling', 'J', TO_DATE('1965-07-31', 'YYYY-MM-DD')); INSERT INTO AUTHOR VALUES (AUTHOR_SEQ.NEXTVAL, 'Jane', 'Austen', 'J', TO_DATE('1775-12-16', 'YYYY-MM-DD')); INSERT INTO AUTHOR VALUES (AUTHOR_SEQ.NEXTVAL, 'Frank', 'Herbert', 'F', TO_DATE('1920-10-08', 'YYYY-MM-DD')); INSERT INTO AUTHOR VALUES (AUTHOR_SEQ.NEXTVAL, 'Walter', 'Isaacson', 'W', TO_DATE('1952-05-20', 'YYYY-MM-DD')); INSERT INTO AUTHOR VALUES (AUTHOR_SEQ.NEXTVAL, 'F. Scott', 'Fitzgerald', 'F', TO_DATE('1896-09-24', 'YYYY-MM-DD')); INSERT INTO AUTHOR VALUES (AUTHOR_SEQ.NEXTVAL, 'David', 'Gemmell', 'D', TO_DATE('1948-08-01', 'YYYY-MM-DD'));

<table>
<tr>
<td></td>
<td>
INSERT INTO AUTHOR
VALUES (AUTHOR_SEQ.NEXTVAL, 'Robert', 'Jordan', 'R', TO_DATE('1948-10-17', 'YYYY-MM-DD'));

Output example:
<table>
<tr><th></th><th>AUTH_ID</th><th>AUTH_FNAME</th><th>AUTH_LNAME</th><th>AUTH_INIT</th><th>AUTH_DOB</th></tr>
<tr><td>1</td><td>1</td><td>J.K.</td><td>Rowling</td><td>J</td><td>31/JUL/65</td></tr>
<tr><td>2</td><td>2</td><td>Jane</td><td>Austen</td><td>J</td><td>16/DEC/75</td></tr>
<tr><td>3</td><td>3</td><td>Frank</td><td>Herbert</td><td>F</td><td>08/OCT/20</td></tr>
<tr><td>4</td><td>4</td><td>Walter</td><td>Isaacson</td><td>W</td><td>20/MAY/52</td></tr>
<tr><td>5</td><td>5</td><td>F. Scott</td><td>Fitzgerald</td><td>F</td><td>24/SEP/96</td></tr>
<tr><td>6</td><td>6</td><td>David</td><td>Gemmell</td><td>D</td><td>01/AUG/48</td></tr>
<tr><td>7</td><td>7</td><td>Robert</td><td>Jordan</td><td>R</td><td>17/OCT/48</td></tr>
</table>
</td>
</tr>
<tr>
<td>BOOK_AUTHOR</td>
<td>
INSERT INTO BOOK_AUTHOR
VALUES (1, 5);

INSERT INTO BOOK_AUTHOR
VALUES (2, 1);

INSERT INTO BOOK_AUTHOR
VALUES (3, 2);

INSERT INTO BOOK_AUTHOR
VALUES (4, 3);

INSERT INTO BOOK_AUTHOR
VALUES (5, 4);

INSERT INTO BOOK_AUTHOR
VALUES (6, 7);

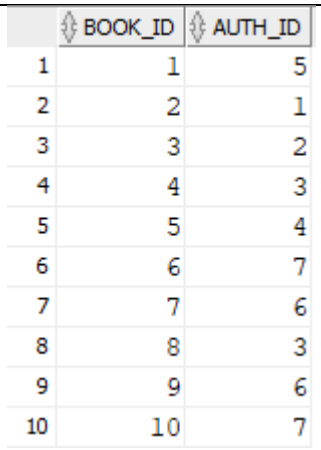
INSERT INTO BOOK_AUTHOR
VALUES (7, 6);

INSERT INTO BOOK_AUTHOR
VALUES (8, 3);

INSERT INTO BOOK_AUTHOR
VALUES (9, 6);

INSERT INTO BOOK_AUTHOR
VALUES (10, 7);

Output example:
</td>
</tr>
</table>

	 BOOK_ID AUTH_ID 1 1 5 2 2 1 3 3 2 4 4 3 5 5 4 6 6 7 7 7 6 8 8 3 9 9 6 10 10 7
STAFF LIBRARIAN TECHNICAL_STAFF SUPPORT_STAFF	INSERT INTO STAFF VALUES (STAFF_SEQ.NEXTVAL, 'John', 'Doe', '1234567890156', '123-456-7890', 'T'); INSERT INTO TECHNICAL_STAFF VALUES (STAFF_SEQ.CURRVAL, 5000.00, 10.0); INSERT INTO STAFF VALUES (STAFF_SEQ.NEXTVAL, 'Jane', 'Smith', '9876543210453', '987-654-3210', 'T'); INSERT INTO TECHNICAL_STAFF VALUES (STAFF_SEQ.CURRVAL, 4500.00, 8.0); INSERT INTO STAFF VALUES (STAFF_SEQ.NEXTVAL, 'David', 'Johnson', '5555555555555', '555-555-5555', 'S'); INSERT INTO SUPPORT_STAFF VALUES (STAFF_SEQ.CURRVAL, 20.50, 40.0); INSERT INTO STAFF VALUES (STAFF_SEQ.NEXTVAL, 'Sarah', 'Williams', '7777777777777', '777-777-7777', 'S'); INSERT INTO SUPPORT_STAFF VALUES (STAFF_SEQ.CURRVAL, 19.45, 51.80); INSERT INTO STAFF VALUES (STAFF_SEQ.NEXTVAL, 'Robert', 'Brown', '8888888888888', '888-888-8888', 'L'); INSERT INTO LIBRARIAN VALUES (STAFF_SEQ.CURRVAL, 'librarian1', 'password123', 1000.00, 'Y'); INSERT INTO STAFF VALUES (STAFF_SEQ.NEXTVAL, 'Emily', 'Jones', '9999999999999', '999-999-9999', 'L'); INSERT INTO LIBRARIAN VALUES (STAFF_SEQ.CURRVAL, 'librarian2', 'password456', 950.00, 'N'); Output examples: STAFF TABLE

	STA...	STAFF_FNAME	STAFF_LNAME	STAFF_IDNO	STAFF_PHONE	STAFF_TYPE
1	1	John	Doe	1234567890156	123-456-7890	T
2	2	Jane	Smith	9876543210453	987-654-3210	T
3	3	David	Johnson	5555555555555	555-555-5555	S
4	4	Sarah	Williams	7777777777777	777-777-7777	S
5	5	Robert	Brown	8888888888888	888-888-8888	L
6	6	Emily	Jones	9999999999999	999-999-9999	L

SUPPORT_STAFF TABLE

	STAFF_ID	PAY_PER_HR	HRS_WORKED
1	3	20.5	40
2	4	19.45	51.8

TECHNICAL STAFF TABLE

	STAFF_ID	SALARY	DATA_ALLOWANCE
1	1	5000	10
2	2	4500	8

LIBRARIAN TABLE

	STAFF_ID	LIB_USERNAME	LIB_PASSWORD	WEEKLY_WAGE	IS_ADMIN
1	5	librarian1	password123	1000	Y
2	6	librarian2	password456	950	N

RENTAL

```
INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 1, TO_DATE('2023-05-20', 'YYYY-MM-DD'), 6);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 2, TO_DATE('2023-05-21', 'YYYY-MM-DD'), 5);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 3, TO_DATE('2023-05-22', 'YYYY-MM-DD'), 6);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 4, TO_DATE('2023-05-23', 'YYYY-MM-DD'), 5);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 5, TO_DATE('2023-05-24', 'YYYY-MM-DD'), 6);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 4, TO_DATE('2023-05-19', 'YYYY-MM-DD'), 5);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 3, TO_DATE('2023-05-20', 'YYYY-MM-DD'), 6);

INSERT INTO RENTAL
VALUES (RENTAL_SEQ.NEXTVAL, 4, TO_DATE('2023-05-18', 'YYYY-MM-DD'), 6);

INSERT INTO RENTAL
```

	VALUES (RENTAL_SEQ.NEXTVAL, 5, TO_DATE('2023-05-21', 'YYYY-MM-DD'), 5); INSERT INTO RENTAL VALUES (RENTAL_SEQ.NEXTVAL, 1, TO_DATE('2023-05-17', 'YYYY-MM-DD'), 5); INSERT INTO RENTAL VALUES (RENTAL_SEQ.NEXTVAL, 2, TO_DATE('2023-05-22', 'YYYY-MM-DD'), 6); INSERT INTO RENTAL VALUES (RENTAL_SEQ.NEXTVAL, 3, TO_DATE('2023-05-16', 'YYYY-MM-DD'), 6); Output example: RENTAL_ID MEM_ID RENTAL_DATE LIB_ID 1 1 1 20/MAY/23 6 2 2 2 21/MAY/23 5 3 3 3 22/MAY/23 6 4 4 4 23/MAY/23 5 5 5 5 24/MAY/23 6 6 6 4 19/MAY/23 5 7 7 3 20/MAY/23 6 8 8 4 18/MAY/23 6 9 9 5 21/MAY/23 5 10 10 1 17/MAY/23 5 11 11 2 22/MAY/23 6 12 12 3 16/MAY/23 6
RENTAL_DETAIL	INSERT INTO RENTAL_DETAIL VALUES (2, 1, SYSDATE, 'Y'); INSERT INTO RENTAL_DETAIL VALUES (2, 2, SYSDATE, 'Y'); INSERT INTO RENTAL_DETAIL VALUES (3, 3, SYSDATE, 'Y'); INSERT INTO RENTAL_DETAIL VALUES (3,4, SYSDATE, 'Y'); INSERT INTO RENTAL_DETAIL VALUES (4,5, SYSDATE, 'Y'); INSERT INTO RENTAL_DETAIL VALUES (1,6, SYSDATE, 'N'); INSERT INTO RENTAL_DETAIL VALUES (5,2, SYSDATE, 'N'); INSERT INTO RENTAL_DETAIL VALUES (6,7, SYSDATE, 'Y'); INSERT INTO RENTAL_DETAIL

<table>
<tr><td></td><td>VALUES (7,3, SYSDATE, 'Y');

INSERT INTO RENTAL_DETAIL
VALUES (8,2, SYSDATE, 'Y');

INSERT INTO RENTAL_DETAIL
VALUES (9,8, SYSDATE, 'N');

INSERT INTO RENTAL_DETAIL
VALUES (10,4, SYSDATE, 'Y');

Output example:
<table>
<tr><th></th><th>RENTAL_ID</th><th>BOOK_ID</th><th>RENTAL_RETURN_DATE</th><th>RENTAL_RETURNED</th></tr>
<tr><td>1</td><td>2</td><td>1</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>2</td><td>2</td><td>2</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>3</td><td>3</td><td>3</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>4</td><td>3</td><td>4</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>5</td><td>4</td><td>5</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>6</td><td>1</td><td>6</td><td>24/MAY/23</td><td>N</td></tr>
<tr><td>7</td><td>5</td><td>2</td><td>24/MAY/23</td><td>N</td></tr>
<tr><td>8</td><td>6</td><td>7</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>9</td><td>7</td><td>3</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>10</td><td>8</td><td>2</td><td>24/MAY/23</td><td>Y</td></tr>
<tr><td>11</td><td>9</td><td>8</td><td>24/MAY/23</td><td>N</td></tr>
<tr><td>12</td><td>10</td><td>4</td><td>24/MAY/23</td><td>Y</td></tr>
</table>
</td></tr>
<tr><td>BOOK_FEE</td><td>INSERT INTO BOOK_FEE
VALUES (BOOK_FEE_SEQ.nextval, 2, 1, 10.99, 'Y');

INSERT INTO BOOK_FEE
VALUES (BOOK_FEE_SEQ.nextval, 3, 3, 5.99, 'N');

Output example:
<table>
<tr><th></th><th>FEE_NUM</th><th>RENTAL_ID</th><th>BOOK_ID</th><th>FEE_AMOUNT</th><th>FEE_PAID</th></tr>
<tr><td>1</td><td>1</td><td>2</td><td>1</td><td>10.99</td><td>Y</td></tr>
<tr><td>2</td><td>2</td><td>3</td><td>3</td><td>5.99</td><td>N</td></tr>
</table>
</td></tr>
<tr><td>RESERVATION</td><td>INSERT INTO RESERVATION
VALUES (1, 1, TO_DATE('2023-05-23', 'YYYY-MM-DD'), 5);

INSERT INTO RESERVATION
VALUES (2, 3, TO_DATE('2023-05-23', 'YYYY-MM-DD'), 5);

INSERT INTO RESERVATION
VALUES (3, 2, TO_DATE('2023-05-24', 'YYYY-MM-DD'), 5);

INSERT INTO RESERVATION
VALUES (4, 5, TO_DATE('2023-05-24', 'YYYY-MM-DD'), 6);

INSERT INTO RESERVATION
VALUES (5, 4, TO_DATE('2023-05-25', 'YYYY-MM-DD'), 6);</td></tr>
</table>

Output example:

	MEM_ID	BOOK_ID	RESERVATIONDATE	LIB_ID
1	1	1	23/MAY/23	5
2	2	3	23/MAY/23	5
3	3	2	24/MAY/23	5
4	4	5	24/MAY/23	6
5	5	4	25/MAY/23	6

iv. Views

BOOK_DETAILS VIEW

```
CREATE OR REPLACE VIEW BOOK_DETAILS AS

SELECT B.BOOK_ID, B.BOOK_TITLE, A.AUTH_FNAME || ' ' || A.AUTH_LNAME AS
AUTHOR, P.PUBLISHER_NAME, G.PRIMARY_GENRE_NAME

FROM BOOK B

JOIN BOOK_AUTHOR BA ON B.BOOK_ID = BA.BOOK_ID

JOIN AUTHOR A ON BA.AUTH_ID = A.AUTH_ID

JOIN PUBLISHER P ON B.PUBLISHER_ID = P.PUBLISHER_ID

JOIN PRIMARY_GENRE G ON B.PRIMARY_GENRE_ID = G.PRIMARY_GENRE_ID;
```

This view includes the following columns: "BOOK_ID," "BOOK_TITLE," "AUTHOR," "PUBLISHER_NAME," and "PRIMARY_GENRE_NAME." Each column represents a specific piece of information related to a book.

The "BOOK_DETAILS" view is helpful because it consolidates relevant data from multiple tables into a single view. By using this view, users can easily access and retrieve comprehensive information about each book, including its unique identifier, title, author's full name, publisher's name, and primary genre. This simplifies querying and reporting tasks by eliminating the need to join multiple tables manually each time the information is required.

Output Example:

	BOOK...	BOOK_TITLE	AUTHOR	PUBLISHER_NAME	PRIMARY_GENRE_NAME
1	1	The Great Gatsby	F. Scott Fitzgerald	Penguin Books	Fiction
2	2	Harry Potter and the Sorcerer's Stone	J.K. Rowling	HarperCollins	Fiction
3	3	Pride and Prejudice	Jane Austen	Random House	Romance
4	4	Dune	Frank Herbert	Simon and Schuster	Science Fiction
5	5	Steve Jobs	Walter Isaacson	Macmillan Publishers	Biography
6	6	The Eye of the World	Robert Jordan	HarperCollins	Fantasy
7	7	Bloodstone	David Gemmell	Simon and Schuster	Fantasy
8	8	Destination: Void	Frank Herbert	Macmillan Publishers	Science Fiction
9	9	Legend	David Gemmell	Random House	Fantasy
10	10	The Great Hunt	Robert Jordan	Penguin Books	Fantasy

CURRENT_RENTALS VIEW

```
CREATE OR REPLACE VIEW CURRENT_RENTALS AS

SELECT R.RENTAL_ID, M.MEM_FNAME || ' ' || M.MEM_LNAME AS MEMBER,
B.BOOK_TITLE, RD.RENTAL_RETURN_DATE

FROM RENTAL R

JOIN MEMBER M ON R.MEM_ID = M.MEM_ID

JOIN RENTAL_DETAIL RD ON R.RENTAL_ID = RD.RENTAL_ID

JOIN BOOK B ON RD.BOOK_ID = B.BOOK_ID

WHERE RD.RENTAL_RETURNED = 'N';
```

The view includes the following columns: "RENTAL_ID," "MEMBER," "BOOK_TITLE," and "RENTAL_RETURN_DATE." Each column represents specific information related to a current book rental.

The "CURRENT_RENTALS" view is helpful because it presents a consolidated view of the current book rentals. By utilizing this view, users can easily access and retrieve relevant information about each rental, including the rental ID, the member's full name, the title of the rented book, and the expected return date.

The view is especially useful for library staff or administrators who need to monitor and manage the current book rentals. It provides an overview of active rentals, allowing them to track which books are currently borrowed by which members.

Output example:

	RENTAL_ID	MEMBER	BOOK_TITLE	RENTAL_RETURN_DATE
1	1	John Doe	The Eye of the World	24/MAY/23
2	9	David Brown	Destination: Void	24/MAY/23
3	5	David Brown	Harry Potter and the Sorcerer's Stone	24/MAY/23

STAFF_INFORMATION VIEW

The "STAFF_INFORMATION" view provides comprehensive information about staff members in the library, including their ID, full name, staff type, librarian-specific details (username), and technical and support staff details (salary and pay per hour, respectively). The view combines data from multiple tables to present a consolidated and organized view of staff information.

This view is helpful as it allows easy access to key details about staff members in a single query, eliminating the need to join multiple tables manually. It provides a concise overview of staff information, facilitating tasks such as generating reports, analysing staff data, and gaining insights into the composition and roles of the library's workforce.

```
CREATE OR REPLACE VIEW STAFF_INFORMATION AS

SELECT S.STAFF_ID, S.STAFF_FNAME || ' ' || S.STAFF_LNAME AS STAFF_NAME,
S.STAFF_TYPE,

    L.LIB_USERNAME, L.LIB_PASSWORD, L.IS_ADMIN, TS.SALARY, SS.PAY_PER_HR

FROM STAFF S

LEFT JOIN LIBRARIAN L

  ON S.STAFF_ID = L.STAFF_ID

LEFT JOIN TECHNICAL_STAFF TS

  ON S.STAFF_ID = TS.STAFF_ID

LEFT JOIN SUPPORT_STAFF SS

  ON S.STAFF_ID = SS.STAFF_ID;
```

Output example:

	STAFF_ID	STAFF_NAME	STAFF_TYPE	LIB_USERNAME	LIB_PASSWORD	IS_ADMIN	SALARY	PAY_PER_HR
1	1	John Doe	T	(null)	(null)	(null)	5000	(null)
2	2	Jane Smith	T	(null)	(null)	(null)	4500	(null)
3	3	David Johnson	S	(null)	(null)	(null)	(null)	20.5
4	4	Sarah Williams	S	(null)	(null)	(null)	(null)	19.45
5	5	Robert Brown	L	librarian1	password123	Y	(null)	(null)
6	6	Emily Jones	L	librarian2	password456	N	(null)	(null)

MEMBER_INFORMATION VIEW

```
CREATE OR REPLACE VIEW MEMBER_INFORMATION AS

SELECT M.MEM_FNAME || ' ' || M.MEM_LNAME AS MEMBER_NAME,
M.MEM_PHONE, M.MEM_STATUS,

    SUM(CASE WHEN RD.RENTAL_RETURNED = 'N' THEN 1 ELSE 0 END) AS "Books
Borrowed",

    SUM(BF.FEE_AMOUNT) AS "Total Amount Owed"

FROM MEMBER M

LEFT JOIN RENTAL R

  ON M.MEM_ID = R.MEM_ID

LEFT JOIN RENTAL_DETAIL RD

  ON RD.RENTAL_ID = R.RENTAL_ID

LEFT JOIN BOOK_FEE BF

  ON RD.RENTAL_ID = BF.RENTAL_ID

GROUP BY M.MEM_FNAME, M.MEM_LNAME, M.MEM_PHONE,
M.MEM_STATUS;
```

The "MEMBER_INFORMATION" view provides a consolidated and organized view of member information in the library. It combines data from multiple tables to present key details about library members in a single query, eliminating the need for manual joins.

The view includes the member's full name, phone number, membership status, the number of books borrowed by the member (excluding returned books), and the total amount owed by the member in terms of book fees.

By joining the "MEMBER," "RENTAL," "RENTAL_DETAIL," and "BOOK_FEE" tables, the view calculates the number of books borrowed by each member and the total amount owed, considering only those books that have not been returned.

This view is beneficial for accessing comprehensive information about library members, facilitating tasks such as generating reports, analysing member data, and gaining insights into borrowing patterns and outstanding fees. It provides a concise overview of member information, enabling efficient management of library operations and member services.

Output example:

	MEMBER_NAME	MEM_PHONE	MEM_STATUS	Books Borrowed	Total Amount Owed
1	Jane Smith	555-5678	Inactive	1	21.98
2	John Doe	555-1234	Active	0	(null)
3	Sarah Williams	555-7890	Active	1	(null)
4	Michael Johnson	555-2468	Active	1	11.98
5	David Brown	555-1357	Inactive	0	(null)

2. Queries

Based on the information required for Untold Stories, the following database functionality has been identified:

i. Query 1: Get Inventory/Book Information

It was stated that the library needs access to the information regarding the library's book inventory, including the title, author, genre, publisher, ISBN, date of publication, number of copies available, the number of copies currently on loan, the number of books damaged, and the number of books that are lost. The following query will be used to help the librarians gain access to this information:

```
SELECT
   B.BOOK_TITLE AS "Title",
   A.AUTH_FNAME || ' ' || A.AUTH_LNAME AS "Author",
   G.PRIMARY_GENRE_NAME AS "Genre",
   P.PUBLISHER_NAME AS "Publisher",
   B.BOOK_ISBN AS "ISBN",
   B.BOOK_RELDATE AS "Date of Publication",
   COUNT(B.BOOK_ID) AS "Total Copies",
   SUM(CASE WHEN RD.RENTAL_RETURNED = 'N' THEN 1 ELSE 0 END) AS "Copies
On Loan",
   SUM(CASE WHEN B.IN_CIRCULATION = 'N' THEN 1 ELSE 0 END) AS "Copies Lost"
FROM BOOK B
LEFT JOIN BOOK_AUTHOR BA ON B.BOOK_ID = BA.BOOK_ID
LEFT JOIN AUTHOR A ON BA.AUTH_ID = A.AUTH_ID
LEFT JOIN PRIMARY_GENRE G ON B.PRIMARY_GENRE_ID =
G.PRIMARY_GENRE_ID
LEFT JOIN PUBLISHER P ON B.PUBLISHER_ID = P.PUBLISHER_ID
LEFT JOIN RENTAL_DETAIL RD ON B.BOOK_ID = RD.BOOK_ID
GROUP BY
   B.BOOK_TITLE,
   A.AUTH_FNAME,
   A.AUTH_LNAME,
   G.PRIMARY_GENRE_NAME,
   P.PUBLISHER_NAME,
   B.BOOK_ISBN,
   B.BOOK_RELDATE;
```

Output example:

	Title	Author	Genre	Publisher	ISBN	Date of Pu...	Total Copies	Copies On Loan	Copies Lost
1	Harry Potter and t...	J.K. Rowling	Fiction	HarperCollins	9780590353427	01/OCT/01	3	1	0
2	Bloodstone	David Gemmell	Fantasy	Simon and Schuster	1534856212579	29/OCT/97	1	0	0
3	The Great Gatsby	F. Scott Fi...	Fiction	Penguin Books	9780743273565	15/JAN/22	1	0	0
4	Dune	Frank Herbert	Science ...	Simon and Schuster	9780441172719	01/JUN/65	2	0	0
5	Destination: Void	Frank Herbert	Science ...	Macmillan Publishers	8456321856375	15/JUN/66	1	1	0
6	Pride and Prejudice	Jane Austen	Romance	Random House	9780141439518	28/JAN/13	2	0	0
7	Steve Jobs	Walter Isaa...	Biography	Macmillan Publishers	9781451648539	24/OCT/11	1	0	0
8	Legend	David Gemmell	Fantasy	Random House	6485123578426	27/MAR/84	1	0	1
9	The Eye of the World	Robert Jordan	Fantasy	HarperCollins	0312850093513	15/JAN/90	1	1	0
10	The Great Hunt	Robert Jordan	Fantasy	Penguin Books	7512964822652	15/NOV/90	1	0	1

ii. Query 2: Reporting Information

Untold Stories also need access to the data regarding the outstanding books per period as well as the top ten most popular books per period (e.g., for the whole year of 2023). The following queries will be used to satisfy this:

```
/*==========================================
Get info on books outstanding
==========================================*/
SELECT
   B.BOOK_TITLE,
   COUNT(RD.BOOK_ID) AS "Outstanding Count"
FROM
   RENTAL_DETAIL RD
LEFT JOIN
   BOOK B ON RD.BOOK_ID = B.BOOK_ID
WHERE
   RD.RENTAL_RETURN_DATE < CURRENT_DATE
   AND RD.RENTAL_RETURNED = 'N'
GROUP BY
   B.BOOK_TITLE
ORDER BY
   COUNT(RD.BOOK_ID) DESC;
```

Output example:

	BOOK_TITLE	Outstanding Count
1	Destination: Void	1
2	The Eye of the World	1
3	Harry Potter and the Sorcerer's Stone	1

```
/*=========================================

Get info on top ten books

=========================================*/

SELECT

   B.BOOK_TITLE,

   COUNT(RD.BOOK_ID) AS "Loan Count"

FROM

   RENTAL_DETAIL RD

LEFT JOIN

   BOOK B ON RD.BOOK_ID = B.BOOK_ID

JOIN

   RENTAL R ON RD.RENTAL_ID = R.RENTAL_ID

WHERE

   R.RENTAL_DATE BETWEEN TO_DATE('2023-01-01', 'YYYY-MM-DD') AND
TO_DATE('2023-12-31', 'YYYY-MM-DD')

   AND ROWNUM <=10

GROUP BY

   B.BOOK_TITLE

ORDER BY

   COUNT(RD.BOOK_ID) DESC;
```

Output example:

	BOOK_TITLE	Loan Count
1	Harry Potter and the Sorcerer's Stone	3
2	Pride and Prejudice	2
3	The Great Gatsby	1
4	The Eye of the World	1
5	Steve Jobs	1
6	Dune	1
7	Bloodstone	1

Note: *Because there are only 7 books currently in the database, only 7 books will display*

iii. Query 3: Member information

To provide librarians with a convenient overview of registered members, the database needs to retrieve the following information: the member's full name, phone number, and membership status. Additionally, it should include the current number of books borrowed by each member and the corresponding outstanding fees they owe. The following SQL query will be used:

```
SELECT * FROM MEMBER_INFORMATION
```

Output example:

	MEMBER_NAME	MEM_PHONE	MEM_STATUS	Books Borrowed	Total Amount Owed
1	Jane Smith	555-5678	Inactive	0	21.98
2	John Doe	555-1234	Active	1	(null)
3	Sarah Williams	555-7890	Active	0	(null)
4	Michael Johnson	555-2468	Active	0	11.98
5	David Brown	555-1357	Inactive	2	(null)

iv. Query 4: Penalty fee information

In order to fulfil the requirement of tracking penalty fees owed by members for late, lost, or damaged books, the following query was utilized:

```
SELECT MEMBER_NAME, "Total Amount Owed"

FROM MEMBER_INFORMATION

WHERE "Total Amount Owed" IS NOT NULL;
```

This query retrieves the names of members and their corresponding total amount owed from the "MEMBER_INFORMATION" view. By using the condition "Total Amount Owed IS NOT NULL" in the query, it filters out any records where no amount is owed, allowing for a focus on the penalty fees that need to be tracked.

Output example:

	MEMBER_NAME	Total Amount Owed
1	Jane Smith	21.98
2	Michael Johnson	11.98

v. Query 5: Librarian information

To address scenarios where a librarian encounters issues with their username, password, or requires retrieval of their personal information, the following query can be employed:

```
SELECT

STAFF_ID, STAFF_NAME, LIB_USERNAME AS "USERNAME", LIB_PASSWORD AS "PASSWORD", IS_ADMIN

FROM STAFF_INFORMATION

WHERE STAFF_TYPE = 'L';
```

This query retrieves relevant details from the "STAFF_INFORMATION" view, specifically the staff ID, staff name, librarian username, password, and administrative privileges. By specifying the condition "STAFF_TYPE = 'L'", only records corresponding to librarians are returned, ensuring that the query focuses on the necessary information for librarians specifically.

Output example:

	STAFF_ID	STAFF_NAME	USERNAME	PASSWORD	IS_ADMIN
1	5	Robert Brown	librarianl	password123	Y
2	6	Emily Jones	librarian2	password456	N

vi. Query 6: Author information

To determine the current number of books written by each author in the library, the following query is utilized:

```
SELECT DISTINCT AUTHOR, COUNT(AUTHOR) AS "Number of books in library"
FROM BOOK_DETAILS
GROUP BY AUTHOR;
```

This query facilitates the calculation of the total count of books authored by each writer presently available in the library. It achieves this by selecting distinct author names from the "BOOK_DETAILS" table and applying the COUNT function to determine the number of occurrences of each author. The result is grouped by author to present a comprehensive overview of the number of books attributed to everyone.

Output example:

	AUTHOR	Number of books in library
1	Frank Herbert	2
2	David Gemmell	2
3	J.K. Rowling	1
4	F. Scott Fitzgerald	1
5	Robert Jordan	2
6	Walter Isaacson	1
7	Jane Austen	1

vii. Query 7: Genre information

To determine the number of books per genre currently in the library we made use of the following query:

```
SELECT DISTINCT PRIMARY_GENRE_NAME, COUNT(PRIMARY_GENRE_NAME) AS
"Number of books in library"

FROM BOOK_DETAILS

GROUP BY PRIMARY_GENRE_NAME;
```

The query retrieves unique primary genre names from the "BOOK_DETAILS" table and calculates the count of books associated with each genre. This information is grouped by the primary genre name and presented as the "Number of books in library" for each genre. The result provides an overview of the distribution of books across different primary genres in the library.

Output example:

	PRIMARY_GENRE_NAME	Number of books in library
1	Fiction	2
2	Biography	1
3	Science Fiction	2
4	Romance	1
5	Fantasy	4

viii. Query 8: Reservation information

The following query retrieves information about reservations made in the library, including the book title, member's first name, and reservation date. It accomplishes this through the following steps:

```
SELECT B.BOOK_TITLE, M.MEM_FNAME, R. RESERVATION_DATE

FROM reservation R

JOIN BOOK B on R.BOOK_ID = B.BOOK_ID

JOIN MEMBER M on R.MEM_ID = M.MEM_ID

WHERE RESERVATION_DATE > SYSDATE;
```

The query combines data from the "reservation," "BOOK," and "MEMBER" tables to retrieve details about reservations in the library. It fetches the book title, member's first name, and reservation date for each reservation record. The query includes join operations to connect the relevant tables based on matching IDs. Additionally, a condition is applied to filter out reservations with a date later than the current system date (SYSDATE). The result provides a list of reservations that are scheduled for a future date.

Output example:

	BOOK_TITLE	MEM_FNAME	RESERVATION_DATE
1	Dune	David	25/MAY/23

ix. Query 9: Inventory Management query

Library staff or administrators might need to quickly locate a book based on a partial title or an ID. The following query enables them to perform a search using a keyword and retrieve the relevant book records.

```
SELECT *

FROM BOOK

WHERE UPPER(BOOK_TITLE) LIKE UPPER('%&Keyword%') OR BOOK_ID LIKE '%&Keyword%';
```

The query searches for books in the library database using a keyword specified by the user. It looks for matches in both the book title and the book ID. The '%' symbol represents a wildcard, allowing the keyword to appear anywhere within the title or ID. The result of the query includes all book records that have a matching title or ID with the provided keyword.

Output example:

	BOOK_ID	BOOK_TITLE	BOOK_ISBN	P...	BOOK_RELDATE	...	BOOK_DESCRIPT	IN_CIRCULATION
1	1	The Great Gatsby	9780743273565	1	15/JAN/22	1	A classic novel by F. Scott Fitzgerald.	Y
2	2	Harry Potter and the So...	9780590353427	2	01/OCT/01	1	The first book in the Harry Potter s...	Y
3	6	The Eye of the World	0312850093513	2	15/JAN/90	6	The first novel in a fantasy series ...	Y
4	10	The Great Hunt	7512964822652	1	15/NOV/90	6	The second novel in a fantasy series...	N

x. Query 10: Support Staff Salary Calculation

The library also needs to calculate the salary to be received by the support staff. To prevent human error, the database does it automatically with this query:

```
SELECT

S.STAFF_FNAME || ' ' || S.STAFF_LNAME AS "Staff member",

ROUND(SS.HRS_WORKED*SS.PAY_PER_HR, 2) AS "Salary"

FROM SUPPORT_STAFF SS

JOIN STAFF S ON SS.STAFF_ID = S.STAFF_ID
```

Output example:

	Staff member	Salary
1	David Johnson	820
2	Sarah Williams	1007.51

xi. Query 11: Finding the most recent transaction

The following query is used to identify the last transaction in our library. This serves as a security measure to ensure that the librarians have visibility on the latest transaction and helps maintain a secure environment. Additionally, it facilitates tasks such as identifying active members, generating rental activity reports, and providing personalized recommendations. By executing this query, the librarians can easily retrieve information about the last person who made a transaction, enabling them to monitor library activity effectively and enhance the overall user experience.

```
SELECT MEM_FNAME, MEM_LNAME
FROM MEMBER
WHERE MEM_ID IN (
  SELECT MEM_ID
  FROM RENTAL
  WHERE RENTAL_DATE >= (SELECT MAX(RENTAL_DATE) FROM RENTAL));
```

Ouput example:

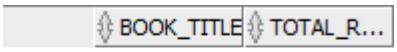
	MEM_FNAME	MEM_LNAME
1	David	Brown

xii. Query 12: Determining the least popular books

The following query generates a report of books that have not been rented in the library, this will help the library determine which books they should not focus on purchasing for their collection:

```
SELECT B.BOOK_TITLE, COUNT(*) AS TOTAL_RENTALS
FROM RENTAL_DETAIL RD
JOIN BOOK B ON RD.BOOK_ID = B.BOOK_ID
GROUP BY B.BOOK_TITLE
HAVING COUNT(*) < 1;
```

Output example:



Note: No results will display based on our example as all our books have been rented out