2024 Spring CS504 Project

Ravi Teja Talluri

G01459601

George Mason University

Database Design

Project Description:

This is the report explaining the creation of a Database Management system for the Public Library. The main purpose of the system is to provide a solution for maintaining the library daily activities related to the customers actions regarding the book borrow and the details about the books and their placement in the library and author details. The scope of the project mainly revolves around library material management and the customer status regarding the material borrowing and returning.

Entities and their relationships:

The material entity contains the details about the Material ID, title of the material and publication date. The Author table contains the Author details like Name, Nationality, Birth date and his ID. The Catalog has the details about the name, place, and its id. Genre gives the information like the materials genre like which type contents in that material and the information the material comes under which category. The borrow table contains details about the borrow id, borrow date, due date and return date. Staff table having the information about the staff details and the member table contains the information about the members details.

1.The Catalog and the Material have one-to-many relationship as a catalog contains the information about many books.

2. One genre contains many books, and each book belongs to one type of genre so there is a one-to-many relationship between the Genre and Material entities.

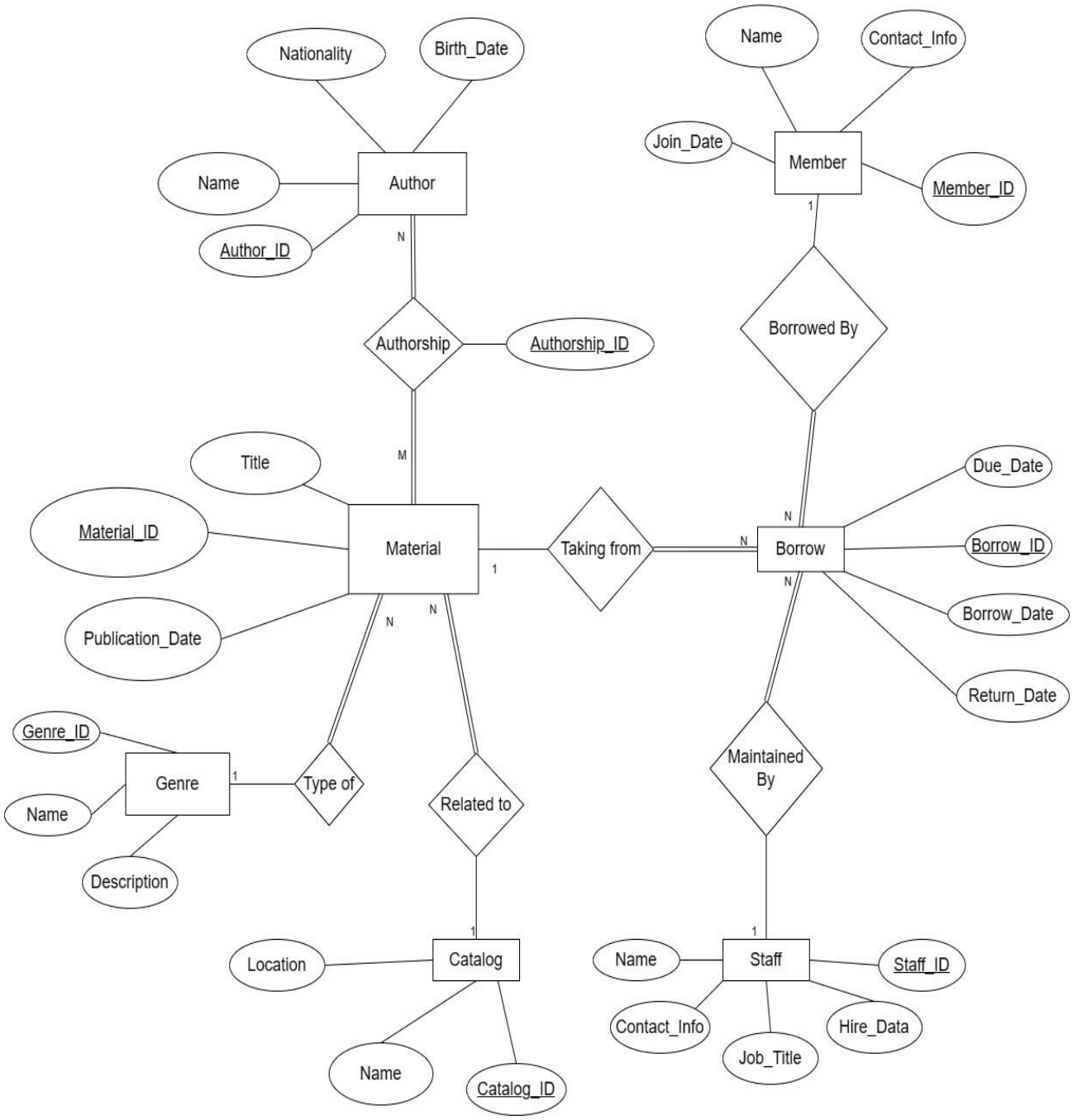
3.Author and Material have many-to-many (n:m) relationship as materials have more than one author and each author has many books.

4.Material and Borrow entities are forming one-to-many(1:n) relationship.

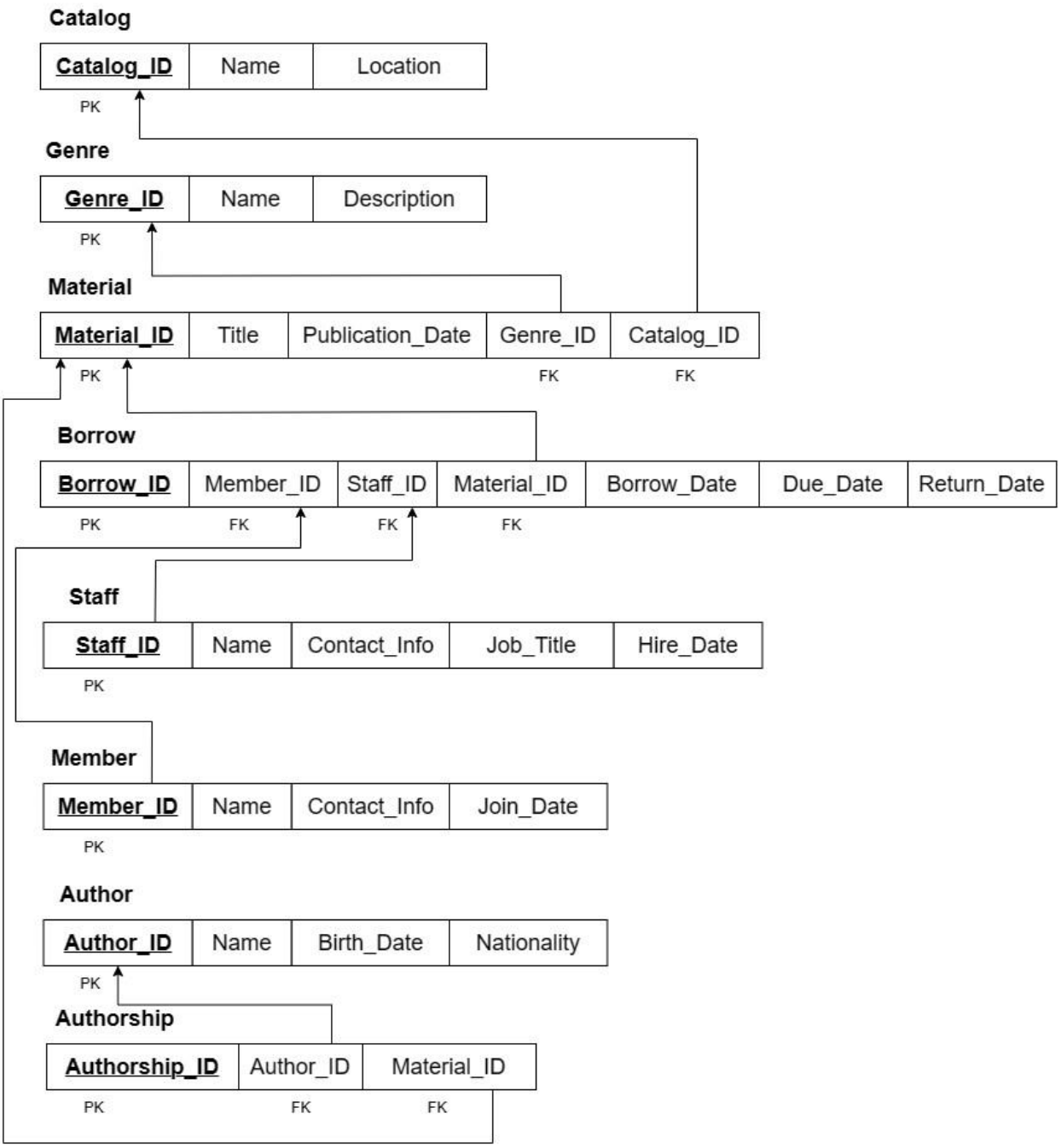
5. The member can borrow many materials and the Member and borrow having one-to-many relationship.

6. The borrowed details are maintained by staff and the multiple records can be maintained by one staff member and those entities are having one-to-many relationship.

ER Diagram:



Relational Schema:



Database Implementation

I used the MySQL Workbench as DBMS for the project, and it is very simple and stable and faster. I created the CSProject Database and implemented the SQL Queries in the MySQL Workbench.

Creating the Database:

```
create Database CSProject;

use  CSProject;
```

SQL Queries for the Table Creation:

```
-- Creating Catalog Table
CREATE TABLE Catalog (
    Catalog_ID INT PRIMARY KEY,
    Name VARCHAR(400),
    Location VARCHAR(400)
    );
```

```sql
-- Creating Genre Table
CREATE TABLE Genre (
    Genre_ID INT PRIMARY KEY,
    Name VARCHAR(400),
    Description VARCHAR(400)
    );

-- Creating Material Table
CREATE TABLE Material (
    Material_ID INT PRIMARY KEY,
    Title VARCHAR(400),
    Publication_Date DATE,
    Catalog_ID INT,
    Genre_ID INT,
    FOREIGN KEY (Catalog_ID) REFERENCES Catalog(Catalog_ID),
    FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
    );
```

```sql

-- Creating Member Table
CREATE TABLE Member (
    Member_ID INT PRIMARY KEY,
    Name VARCHAR(400),
    Contact_Info VARCHAR(400),
    Join_Date DATE
    );

-- Creating Staff Table
CREATE TABLE Staff (
   Staff_ID INT PRIMARY KEY,
   Name VARCHAR(400),
   Contact_Info VARCHAR(400),
   Job_Title VARCHAR(400),
   Hire_Date VARCHAR(400)
   );

```

```sql

-- Creating Borrow Table
CREATE TABLE Borrow (
   Borrow_ID INT PRIMARY KEY,
   Material_ID INT,
   Member_ID INT,
   Staff_ID INT,
   Borrow_Date DATE,
   Due_Date DATE,
   Return_Date DATE,
   FOREIGN KEY (Material_ID) REFERENCES Material(Material_ID),
   FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
   FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID)
   );

```

```sql

-- Creating Author Table
CREATE TABLE Author (
Author_ID INT PRIMARY KEY,
Name VARCHAR(400),
Birth_Date DATE,
Nationality VARCHAR(400)
  );

-- Creating Authorship Table
CREATE TABLE Authorship (
Authorship_ID INT PRIMARY KEY,
Author_ID INT,
Material_ID INT,
FOREIGN KEY (Material_ID) REFERENCES Material(Material_ID),
FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID)
  );

```

Loading the Data into the tables:

At first, I Saved the given sample data files into the location ‘C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Author.csv' and loaded the data from the saved csv files into the tables created in the database using SQL Queries.

```sql
-- Loading into Catalog table
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Catalog.csv'
INTO TABLE Catalog
IGNORE 1 ROWS
(Catalog_ID, Name, Location);

-- loading into Genre table
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Genre.csv'
INTO TABLE Genre
IGNORE 1 ROWS
(Genre_ID, Name, Description);
```

```sql
-- Loading into material table
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Material.csv'
INTO TABLE Material
IGNORE 1 ROWS
(Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID);

-- Loading into member table
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Member.csv'
INTO TABLE Member
IGNORE 1 ROWS
(Member_ID, Name, Contact_Info, Join_Date);

-- Loading into staff table
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Staff.csv'
INTO TABLE Staff
IGNORE 1 ROWS
(Staff_ID, Name, Contact_Info, Job_Title, Hire_Date);
```

```sql
-- loading into Borrow table
INSERT INTO Borrow (Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date) VALUES
(1, 1, 1, 1, '2018-09-12', '2018-10-03', '2018-09-30'),
(2, 2, 2, 1, '2018-10-15', '2018-11-05', '2018-10-29'),
(3, 3, 3, 1, '2018-12-20', '2019-01-10', '2019-01-08'),
(4, 4, 4, 1, '2019-03-11', '2019-04-01', '2019-03-27'),
(5, 5, 5, 1, '2019-04-20', '2019-05-11', '2019-05-05'),
(6, 6, 6, 1, '2019-07-05', '2019-07-26', '2019-07-21'),
(7, 7, 7, 1, '2019-09-10', '2019-10-01', '2019-09-25'),
(8, 8, 8, 1, '2019-11-08', '2019-11-29', '2019-11-20'),
(9, 9, 9, 1, '2020-01-15', '2020-02-05', '2020-02-03'),
(10, 10, 10, 1, '2020-03-12', '2020-04-02', '2020-03-28'),
(11, 1, 11, 2, '2020-05-14', '2020-06-04', '2020-05-28'),
(12, 2, 12, 2, '2020-07-21', '2020-08-11', '2020-08-02'),
(13, 3, 13, 2, '2020-09-25', '2020-10-16', '2020-10-15'),
(14, 4, 1, 2, '2020-11-08', '2020-11-29', '2020-11-24'),
(15, 5, 2, 2, '2021-01-03', '2021-01-24', '2021-01-19'),
(16, 6, 3, 2, '2021-02-18', '2021-03-11', '2021-03-12'),
(17, 17, 4, 2, '2021-04-27', '2021-05-18', '2021-05-20'),
(18, 18, 5, 2, '2021-06-13', '2021-07-04', '2021-06-28'),
(19, 19, 6, 2, '2021-08-15', '2021-09-05', '2021-09-03'),
(20, 20, 7, 2, '2021-10-21', '2021-11-11', NULL),
(21, 21, 1, 3, '2021-11-29', '2021-12-20', NULL),
(22, 22, 2, 3, '2022-01-10', '2022-01-31', '2022-01-25'),
(23, 23, 3, 3, '2022-02-07', '2022-02-28', '2022-02-23'),
```

```sql
-- loading into author table
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Author.csv'
INTO TABLE Author
IGNORE 1 ROWS
(Author_ID, Name, Birth_Date, Nationality);

-- loading into authorship
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Authorship.csv'
INTO TABLE Authorship
IGNORE 1 ROWS
(Authorship_ID, Author_ID, Material_ID);
```

Querying and Manipulation

Set of SQL Queries:

SQL Queries for Common Tasks:

Select operator is used to get the data from the required table. I selected the entire staff table for viewing purposes.

```
SELECT * FROM Staff;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Staff_ID	Name	Contact_Info	Job_Title	Hire_Date
1	Amy Green	amy.green@email.com	Librarian	2017-06-01
2	Brian Taylor	brian.taylor@email.com	Library Assistant	2018-11-15
3	Christine King	chris.king@email.com	Library Assistant	2019-05-20
4	Daniel Wright	dan.wright@email.com	Library Technician	2020-02-01
NULL	NULL	NULL	NULL	NULL

UPDATE query is used to update or change the information in the table. Updating the staff table contact info column value where the contact id is 4 and the updating was @email.com to @gmail.com.

```
UPDATE Staff
SET Contact_Info = REPLACE(Contact_Info, '@email.com', '@gmail.com') WHERE Staff_ID=4;
SELECT * FROM Staff;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Staff_ID	Name	Contact_Info	Job_Title	Hire_Date
1	Amy Green	amy.green@email.com	Librarian	2017-06-01
2	Brian Taylor	brian.taylor@email.com	Library Assistant	2018-11-15
3	Christine King	chris.king@email.com	Library Assistant	2019-05-20
4	Daniel Wright	dan.wright@gmail.com	Library Technician	2020-02-01
NULL	NULL	NULL	NULL	NULL

Insert command is used to include or add new values to the table. A new row with details added to the Catalog table.

```
-- Insert command
INSERT INTO Catalog (Catalog_ID, Name, Location)
VALUES (11, 'Ravi', 'K11.1');
SELECT * FROM Catalog WHERE Name = 'Ravi';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Catalog_ID	Name	Location
11	Ravi	K11.1
NULL	NULL	NULL

Delete command. It is mainly used to delete the rows in the table. Here the delete command is used to delete the new row added in the Catalog table.

```
-- Delete Command
DELETE FROM Catalog WHERE Catalog_ID=11;
```

Output

Action Output

#	Time	Action	Message
145	00:52:59	INSERT INTO Catalog (Catalog_ID, Name, Location) VALUES (11, 'Ravi', 'K11.1')	1 row(s) affected
146	00:53:55	SELECT * FROM Catalog WHERE Name = 'Ravi'	1 row(s) returned
147	01:00:19	DELETE FROM Catalog WHERE Name='Ravi'	Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE
148	01:00:44	DELETE FROM Catalog WHERE Catalog_ID=11	1 row(s) affected

Here is the result after executing the delete command.

```
-- Delete Command
DELETE FROM Catalog WHERE Catalog_ID=11;
SELECT * FROM Catalog Where Catalog_ID=11;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Catalog_ID	Name	Location
NULL	NULL	NULL

Catalog 79

Output

Action Output

#	Time	Action	Message
146	00:53:55	SELECT * FROM Catalog WHERE Name = 'Ravi'	1 row(s) returned
147	01:00:19	DELETE FROM Catalog WHERE Name='Ravi'	Error Code: 1175. You are using safe
148	01:00:44	DELETE FROM Catalog WHERE Catalog_ID=11	1 row(s) affected
149	01:03:13	SELECT * FROM Catalog Where Catalog_ID=11	0 row(s) returned

Join condition used to join two tables by the common key column on both tables and there by the required columns from both tables can be executed. The following join command is used to print the list of materials and their genre by joining the Genre Id.

```
-- Join Command
SELECT M.Material_ID, M.Title, G.Name FROM Material M
JOIN Genre G ON M.Genre_ID = G.Genre_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Material_ID	Title	Name
1	The Catcher in the Rye	General Fiction
2	To Kill a Mockingbird	General Fiction
6	Pride and Prejudice	General Fiction
7	The Great Gatsby	General Fiction
8	Moby Dick	General Fiction
9	Crime and Punishment	General Fiction
16	The Adventures of Huckleberry Finn	General Fiction
17	The Catch-22	General Fiction
18	The Picture of Dorian Gray	General Fiction
22	A Tale of Two Cities	General Fiction
25	The Brothers Karamazov	General Fiction
27	The Grapes of Wrath	General Fiction
28	The Old Man and the Sea	General Fiction
29	The Count of Monte Cristo	General Fiction
3	The Da Vinci Code	Mystery & Thr...
32	New book	Mystery & Thr...
4	The Hobbit	Science Fictio...
10	The Hitchhiker's Guide to the Galaxy	Science Fictio...
15	The Chronicles of Narnia: The Lion ...	Science Fictio...

SQL Queries for Advanced Queries are like aggregation. The below query is used to list the Count of Materials based on the Genre type. By joining the Genre table with Material table by using Genre id.

```
-- Aggregation Commands
SELECT  G.Name, COUNT(*) AS TotalCount FROM Genre G JOIN
Material M ON G.Genre_ID = M.Genre_ID GROUP BY G.Name ORDER BY TotalCount DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	TotalCount
General Fiction	14
Science Fiction & Fantasy	4
Horror & Suspense	4
Dystopian & Apocalyptic	3
Classics	3
Mystery & Thriller	2
Historical Fiction	2

Subquery is calling another query in outer query. To retrieve the author name for the material whose id 11 is got by using subquery.

```
-- SubQuery

SELECT Name FROM Author
WHERE Author_ID = (SELECT Author_ID FROM Authorship
                   WHERE Material_ID = (SELECT Material_ID FROM Material WHERE Material_ID = 11));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name
Harper Lee

Q1. Which materials are available in the library?

The available list of materials can be taken from the Material table and by applying condition that the material id is not in the list get by nested query that is querying the list of material id whose return date is null in the borrow list. Thereby it results in the list of items taken by members and in outer query I applied where condition that material id is not in that list. So thereby it results in the list of materials available in the library.

```
-- Q1.Which materials are currently available in the library?
SELECT M.Material_ID, M.Title FROM Material as M
        WHERE Material_ID NOT IN (SELECT Material_ID FROM Borrow WHERE Return_Date IS NULL);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Material_ID	Title
3	The Da Vinci Code
11	1984
12	Animal Farm
13	The Haunting of Hill House
14	Brave New World
15	The Chronicles of Narnia: The Lion the Witch an...
16	The Adventures of Huckleberry Finn
17	The Catch-22
18	The Picture of Dorian Gray
19	The Call of Cthulhu
22	A Tale of Two Cities
23	The Iliad
24	The Odyssey
25	The Brothers Karamazov
26	The Divine Comedy
27	The Grapes of Wrath
28	The Old Man and the Sea
29	The Count of Monte Cristo
30	A Midsummer Night's Dream
31	The Tricky Book
NULL	NULL

Q2. Which materials are currently overdue? Suppose today is 04/01/2023 and show the borrow date and due date of each material.

In this join query is used to join both borrow table and material table based on the material id and thereby applying the condition that the due date is less than the given date.

```
-- Q2.Which materials are currently overdue? Suppose today is 04/01/2023, and show the borrow date and due date of each material.
SELECT m.Material_ID, m.Title, b.Borrow_Date, b.Due_Date FROM Borrow b
JOIN Material m ON b.Material_ID = m.Material_ID WHERE b.Due_Date < '2023-04-01' AND b.Return_Date IS NULL
ORDER BY m.Material_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Material_ID	Title	Borrow_Date	Due_Date
1	The Catcher in the Rye	2022-12-28	2023-01-18
2	To Kill a Mockingbird	2023-01-23	2023-02-13
4	The Hobbit	2023-03-01	2023-03-22
5	The Shining	2023-03-10	2023-03-31
20	Harry Potter and the Philosopher's Stone	2021-10-21	2021-11-11
21	Frankenstein	2021-11-29	2021-12-20

Q3. 10 Topmost Borrowed materials order by their borrow count.

By joining material and borrow table based on material id and the conditions stated as group by material id and the limit of top 10 can result the required list of materials with their borrow count of top 10.

```
-- Q3. What are the top 10 most borrowed materials in the library? Show the title of each
-- material and order them based on their available counts.
SELECT M.Material_ID, M.Title, COUNT(B.Material_ID) as Borrowed_Count FROM Material M
JOIN Borrow B on M.Material_ID = B.Material_ID GROUP BY M.Material_ID
ORDER BY Borrowed_Count DESC LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Material_ID	Title	Borrowed_Count
1	The Catcher in the Rye	3
2	To Kill a Mockingbird	3
3	The Da Vinci Code	3
4	The Hobbit	3
5	The Shining	3
6	Pride and Prejudice	3
7	The Great Gatsby	2
8	Moby Dick	2
9	Crime and Punishment	2
10	The Hitchhiker's Guide to the Galaxy	2

Q4. How many materials has author Lucas Piki written?

The Author and authorship table joined by author id there by authorship table and material table can be joined by material id, so the material list and their respective authors can be printed and getting count of the list of materials can be the required answer for the question.

```sql
-- Q4. How many materials has author Lucas Piki written?
SELECT A.Name, COUNT(*) AS Material_Count FROM Material M
JOIN Authorship AP ON AP.Material_ID = M.Material_ID
JOIN Author A ON AP.Author_ID = A.Author_ID
WHERE A.Name = 'Lucas Piki' GROUP BY A.Name
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Name	Material_Count
Lucas Piki	1

Q5. How many materials were written by two or more authors?

Subquery results the material id by grouping it from authorship table that contains multiple authors count of 2 or more. Outer query counts the results of inner query thereby it gives the count of materials having 2 or more authors.

```sql
-- Q5 How many materials were written by two or more than authors
SELECT COUNT(*) AS CountOfMaterialsWithMultipleAuthors
FROM (
    SELECT Material_ID
    FROM Authorship
    GROUP BY Material_ID
    HAVING COUNT(Author_ID) >= 2
) ASSUBQUERY;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CountOfMaterialsWithMultipleAuthors
4

Q6. What are the most popular genres in the library ranked by the total number of borrowed times of each genre?

By joining the Material and genre by genre id and material with borrow table by material id the relation for genre table and borrow table created and now by querying name and the count of the borrow id will results the no of times borrowed of that genre type by group by the genre id.

```sql
-- Q6 What are the most popular genres in the library ranked by the total number of borrowed
-- times of each genre?
select G.Name as genre_type, COUNT(B.Borrow_ID) AS NoOfTimesBorrowed
FROM Genre G JOIN Material M on G.Genre_ID = M.Genre_ID
JOIN Borrow B ON M.Material_ID = B.Material_ID GROUP BY G.Genre_ID
ORDER BY NoOfTimesBorrowed DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

genre_type	NoOfTimesBorrowed
General Fiction	22
Science Fiction & Fantasy	6
Horror & Suspense	5
Mystery & Thriller	3
Classics	3
Historical Fiction	1

Q7. How many materials had been borrowed from 09/2020-10/2020?

From borrow table the rows count can give the count of no of materials borrowed by applying where condition for the specified date range.

```sql
-- Q7. How many materials had been borrowed from 09/2020-10/2020?
SELECT COUNT(*) AS NoOfMaterialsBorrowed FROM Borrow
Where Borrow_Date BETWEEN '2020-09-01' AND '2020-10-31';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

NoOfMaterialsBorrowed
1

Q8. How do you update the "Harry Poper and the Philosopher's Stone" when it is returned on 04/01/2023?

Writing simple update query for the borrow table return date by selecting material id for the specified material name from the material table in sub query.

```
-- Q8.How do you update the "Harry Poper and the Philosopher's Stone"
-- when it is returned on 04/01/2023?
UPDATE Borrow SET Return_Date = '2023-04-01'
WHERE Material_ID = (SELECT Material_ID FROM Material
WHERE Title = 'Harry Potter and the Philosopher''s Stone');
```

Below query is to check the update statement.

```
-- Query for verifying the Update query
select Return_Date FROM Borrow WHERE
Material_ID = (SELECT Material_ID FROM Material WHERE Title = 'Harry Potter and the Philosopher''s Stone');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Return_Date
2023-04-01

Q9. How do you delete the member Emily Miller and all her related records from the database?

Deleting from borrow table by selecting the member id and by subquery member id can be taken by selecting name in where condition.

```
-- Q9. How do you delete the member Emily Miller
-- and all her related records from the database?
-- Deleting from Borrow table
DELETE FROM Borrow WHERE Member_ID =
(SELECT Member_ID FROM Member WHERE Name = 'Emily Miller');
SELECT * FROM Borrow WHERE Member_ID = 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conten

Borrow_ID	Material_ID	Member_ID	Staff_ID	Borrow_Date	Due_Date	Return_Date
NULL	NULL	NULL	NULL	NULL	NULL	NULL

To remove from the Member table the where condition should use Primary key only in safe mode or we can set safe mode '0' and deleting the record and again we can on the safe mode.

```
223
224 •  SET SQL_SAFE_UPDATES = 0;
225 •  DELETE FROM Member WHERE Name = 'Emily Miller';
226 •  SET SQL_SAFE_UPDATES = 1;
227 •  SELECT * FROM Member WHERE Name = 'Emily Miller'
228
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Borrow_ID	Material_ID	Member_ID	Staff_ID	Borrow_Date	Due_Date	Return_Date
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Q10. How do you add the following material to the database?

Title: New book

Date: 2020-08-01

Catalog: E-Books

Genre: Mystery & Thriller

Author: Lucas Luke

Adding the Values into the table by using Insert statement.

```
-- How do you add the following material to the database?
-- Title: New book, Date: 2020-08-01, Catalog: E-Books, Genre: Mystery & Thriller, Author: Lucas Luke
-- Inserting into Material Table
-- Drop foreign key constraint

INSERT INTO Material (Title, Publication_Date, Catalog_ID, Genre_ID)
VALUES('New book', '2020-08-01',
(SELECT Catalog_ID FROM Catalog WHERE Name = 'E-Books'),
(SELECT Genre_ID FROM Genre WHERE Name = 'Mystery & Thriller'));

-- Inserting into Author table
INSERT INTO Author (Name)
VALUES ('Lucas Luke');

SELECT * FROM Material WHERE Material_ID = 32;
SELECT * FROM Author WHERE Name='Lucas Luke';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Author_ID	Name	Birth_Date	Nationality
21	Lucas Luke	NULL	NULL
NULL	NULL	NULL	NULL

Design of Extended Features

Alert staff about overdue materials on a daily basis?

The stored procedure can be used and scheduled event to run daily specified time to update the procedure daily. The stored procedure here is used to alert the staff about the list of materials and their count of overdue.

```
CREATE PROCEDURE AlertOnOverdueMemberMaterial ()

BEGIN

SELECT M.Material_ID, M.Title, B.Due_Date, B.Member_ID

FROM Borrow B JOIN Material M ON B.Material_ID = M.Material_ID

WHERE B.Return_Date IS NULL AND B.Due_Date < CURDATE();

END;
```

By creating the Event scheduler in MySQL can alert the staff about the overdue by running the above stored procedure timely manner.

```
CREATE EVENT TimelyMannerOverdueAlert

ON SCHEDULE EVERY 1 DAY STARTS '2024-04-15 00:00:00'

DO   CALL AlertOnOverdueMemberMaterial ();
```

Creating procedure to deactivate the membership of overduecount >= 3.

```
CREATE PROCEDURE OverdueDeactivate()

BEGIN

UPDATE Member SET Status = 'Inactive'

WHERE Member_ID IN (SELECT Member_ID FROM Borrow

WHERE Return_Date IS NULL AND Due_Date < CURDATE()

 GROUP BY Member_ID HAVING COUNT(*) >= 3);

END;
```

This Procedure can Deactivate the Membership of the menbers who is Overdue count is Grater than or equal to 3.

But to schedule this event we need to create a event scheduler based upon the requirement like daily or weekly.

Creating Trigger to Reactivate the membership.

Above we used the procedure to deactivate the membership. Now we can use the Procedure to Reactivate the Membership of the member after paying the Overdue fees.

```
CREATE PROCEDURE ReactiveMembership(IN memberID INT)

BEGIN

UPDATE Member SET Status = 'Active'

WHERE Member_ID = memberID;
```

END;

By manual triggering this procedure can Reactivate the membership of the member when he pays the Overdue Fee payment.