(1)

Code-

SELECT Material.Material\_ID, Material.Title

FROM Material

LEFT JOIN Borrow ON Material.Material\_ID = Borrow.Material\_ID AND Borrow.Return\_Date IS NULL

WHERE Borrow.Borrow\_ID IS NULL

ORDER by material\_id;

Explanation-

The SQL query uses an LEFT JOIN to combine the Material and Borrow tables, focusing on records where materials are still borrowed (i.e., Return\_Date is NULL). It then filters to include only materials that do not have active borrow records, effectively listing all materials that are currently available for borrowing due to being returned or never checked out.

A screenshot of a computer

Description automatically generated

(2)

Code –

SELECT Material.Material\_ID, Title,Borrow\_Date,Due\_Date

FROM Borrow

JOIN Material ON Borrow.Material\_ID = Material.Material\_ID

WHERE Borrow.Due\_Date < '2023-04-01' AND (Borrow.Return\_Date IS NULL OR Borrow.Return\_Date > '2023-04-01')

order BY Material\_id;

Explanation –

In this query, I use a join between the Material and Borrow tables based on the Material\_ID to associate each borrowed item with its descriptive title from the Material table. The purpose of this join is to retrieve not only the borrow and due dates from the Borrow table, but also the title of each material from the Material table, giving a comprehensive view of each overdue item. The query specifically filters for materials with due dates prior to April 1, 2023 that have either not yet been returned or were returned after their due date, effectively identifying all overdue materials.

A screenshot of a computer

Description automatically generated

(3)

Code-

SELECT Material.Material\_ID, Title

FROM Material

LEFT JOIN Borrow ON Material.Material\_ID = Borrow.Material\_ID AND Borrow.Return\_Date IS NULL

WHERE Borrow.Borrow\_ID IS NULL

ORDER by material\_id;

Explanation –

This query joins the Material and Borrow tables to associate each borrow record with its corresponding material title. After establishing this connection, it uses the COUNT function to aggregate the total borrow instances for each material, grouping the results by material title to ensure that each title is counted uniquely. The results are then sorted in descending order based on borrow count, and only the top ten materials are displayed, highlighting the library's most popular items.

A screenshot of a computer

Description automatically generated

(4)

Code –

SELECT Author.Name, COUNT(Material.Material\_ID) AS Number\_Of\_Materials

FROM Author

JOIN Authorship ON Author.Author\_ID = Authorship.Author\_ID

JOIN Material ON Authorship.Material\_ID = Material.Material\_ID

WHERE Author.Name = 'Lucas Piki'

GROUP BY Author.Name;

Explanation –

Query begins by connecting the Author, Authorship, and Material tables to ensure that all relevant data about the author, authorship records, and materials are in sync. By including Lucas Piki in the WHERE clause, the query focuses solely on materials related to this author. Use of GROUP BY AUTHOR.Name enables aggregation by author.

A screenshot of a computer

Description automatically generated

(5)

Code-

SELECT Material.Material\_ID, Material.Title, ARRAY\_AGG(Author.Name) AS Author\_Names

FROM Authorship

JOIN Author ON Authorship.Author\_ID = Author.Author\_ID

JOIN Material ON Authorship.Material\_ID = Material.Material\_ID

GROUP BY Material.Material\_ID, Material.Title

HAVING COUNT(DISTINCT Authorship.Author\_ID) > 1;

Explanation-

JOIN Operations: This query joins the Authorship table to the Author and Material tables. This allows you to retrieve the author's name (Author.Name) as well as the material title.

GROUP BY Clause: Groups the results based on Material.Material\_ID and Material.Title to aggregate author names for each unique material.

ARRAY\_AGG: This PostgreSQL function collects all author names into an array for each grouped material, indicating which authors worked together on each work. (This function is specific to PostgreSQL; if you're using another SQL database system, adjust accordingly.)

HAVING COUNT(DISTINCT Author\_ID) > 1: This filters the groups to only include materials with more than one distinct author, which is required for identifying collaborations.

A screenshot of a computer

Description automatically generated

(6)

Code –

SELECT Genre.Name, COUNT(Borrow.Material\_ID) AS Total\_Borrows

FROM Genre

JOIN Material ON Genre.Genre\_ID = Material.Genre\_ID

JOIN Borrow ON Material.Material\_ID = Borrow.Material\_ID

GROUP BY Genre.Name

ORDER BY Total\_Borrows DESC;

Explanation –

I am using Join in this query, JOIN Material By Genre.Material is the genre identifier.Genre\_ID: Connects the Material and Genre tables by using the Genre\_ID to link materials to genres.

Join Borrow ON Material.Material\_ID = Borrow.Material\_ID: Joins the Borrow and Material tables, allowing you to count how many times each material has been borrowed.

GROUP BY Genre.Name: Groups the results by genre name, allowing the COUNT aggregation function to calculate borrows by genre.

ORDER BY Total\_Borrows DESC: Sorts the results by total borrows in descending order, ranking genres from most to least popular based on borrowing activity.

A screenshot of a computer

Description automatically generated

(7)

Code-

SELECT Material.Title,COUNT(Borrow.Material\_ID) AS Borrow\_Count

FROM Material

JOIN Borrow ON Material.Material\_ID = Borrow.Material\_ID

WHERE Borrow.Borrow\_Date BETWEEN '2020-09-01' AND '2020-10-31'

GROUP BY Material.Title

ORDER BY Borrow\_Count DESC;

Explanation –

The query examines the borrowing records for this time period. It identifies each item by name, counts the number of times it has been borrowed, and retrieves the total quantity available in the library. The results are grouped by material to ensure accurate aggregation, and then ordered by number of borrowings, with the most frequently borrowed materials highlighted first.

A screenshot of a computer

Description automatically generated

(8)

Code-

UPDATE Borrow

SET Return\_Date = '2023-04-01'

WHERE Borrow\_ID IN

(SELECT Borrow\_ID

FROM Borrow

JOIN Material ON Borrow.Material\_ID = Material.Material\_ID

WHERE Material.Title = 'Harry Potter and the Philosopher''s Stone' AND Borrow.Return\_Date IS NULL

ORDER BY Borrow.Borrow\_Date DESC

LIMIT 1);

SELECT Borrow.Borrow\_ID, Material.Title, Borrow.Material\_ID, Borrow.Member\_ID, Borrow.Staff\_ID, Borrow.Borrow\_Date, Borrow.Due\_Date, Borrow.Return\_Date

FROM Borrow

JOIN Material ON Borrow.Material\_ID = Material.Material\_ID

WHERE Material.Title = 'Harry Potter and the Philosopher''s Stone' AND Borrow.Return\_Date = '2023-04-01'

ORDER BY Borrow.Borrow\_Date DESC

LIMIT 1;

Explanation - The first part of the code updates the date and second part displays it.

A screenshot of a computer

Description automatically generated

(9)

Code –

DELETE FROM Borrow

WHERE Member\_ID IN (

SELECT Member\_ID

FROM Member

WHERE name = 'Emily Miller'

);

A screenshot of a computer

Description automatically generated

Code –

DELETE FROM Member

WHERE name = 'Emily Miller';

A screenshot of a computer

Description automatically generated

(10)

Code –

SELECT \* From Authors

SELECT Genre\_ID FROM Genre WHERE Name = 'Mystery & Thriller';

INSERT INTO author(author\_id,"name",birth\_date,nationality)

VALUES(21,'Lucas Luke','2020-08-16','American');

INSERT INTO material(material\_id,title,publication\_date,catalog\_id,genre\_id)

VALUES(32, 'New book', '2020-08-01',3,2);

SELECT \* from material

INSERT INTO authorship(authorship\_id,author\_id,material\_id)

VALUES(35,21,32);

Explanation-

First I saw the new author\_id and then I saw the genre\_id for genre “Mystery & Thriller” and Material\_id from material table then,I first inserted the data into Author table. Then into Material table and then lastly into Authoship table.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated