**Library Management Project**

-- CREATE DATABASE library;

use library;

-- Create table "Branch"

DROP TABLE IF EXISTS branch;

CREATE TABLE branch

(

branch\_id VARCHAR(10) PRIMARY KEY,

manager\_id VARCHAR(10),

branch\_address VARCHAR(30),

contact\_no VARCHAR(15)

);

-- Create table "Employee"

DROP TABLE IF EXISTS employees;

CREATE TABLE employees

(

emp\_id VARCHAR(10) PRIMARY KEY,

emp\_name VARCHAR(30),

position VARCHAR(30),

salary DECIMAL(10,2),

branch\_id VARCHAR(10),

FOREIGN KEY (branch\_id) REFERENCES branch(branch\_id)

);

-- Create table "Members"

DROP TABLE IF EXISTS members;

CREATE TABLE members

(

member\_id VARCHAR(10) PRIMARY KEY,

member\_name VARCHAR(30),

member\_address VARCHAR(30),

reg\_date DATE

);

-- Create table "Books"

DROP TABLE IF EXISTS books;

CREATE TABLE books

(

isbn VARCHAR(50) PRIMARY KEY,

book\_title VARCHAR(80),

category VARCHAR(30),

rental\_price DECIMAL(10,2),

status VARCHAR(10),

author VARCHAR(30),

publisher VARCHAR(30)

);

-- Create table "IssueStatus"

DROP TABLE IF EXISTS issued\_status;

CREATE TABLE issued\_status

(

issued\_id VARCHAR(10) PRIMARY KEY,

issued\_member\_id VARCHAR(30),

issued\_book\_name VARCHAR(80),

issued\_date DATE,

issued\_book\_isbn VARCHAR(50),

issued\_emp\_id VARCHAR(10),

FOREIGN KEY (issued\_member\_id) REFERENCES members(member\_id),

FOREIGN KEY (issued\_emp\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (issued\_book\_isbn) REFERENCES books(isbn)

);

-- Create table "ReturnStatus"

DROP TABLE IF EXISTS return\_status;

CREATE TABLE return\_status

(

return\_id VARCHAR(10) PRIMARY KEY,

issued\_id VARCHAR(30),

return\_book\_name VARCHAR(80),

return\_date DATE,

return\_book\_isbn VARCHAR(50),

FOREIGN KEY (return\_book\_isbn) REFERENCES books(isbn)

);

INSERT INTO members(member\_id, member\_name, member\_address, reg\_date)

VALUES

('C101', 'Alice Johnson', '123 Main St', '2021-05-15'),

('C102', 'Bob Smith', '456 Elm St', '2021-06-20'),

('C103', 'Carol Davis', '789 Oak St', '2021-07-10'),

('C104', 'Dave Wilson', '567 Pine St', '2021-08-05'),

('C105', 'Eve Brown', '890 Maple St', '2021-09-25'),

('C106', 'Frank Thomas', '234 Cedar St', '2021-10-15'),

('C107', 'Grace Taylor', '345 Walnut St', '2021-11-20'),

('C108', 'Henry Anderson', '456 Birch St', '2021-12-10'),

('C109', 'Ivy Martinez', '567 Oak St', '2022-01-05'),

('C110', 'Jack Wilson', '678 Pine St', '2022-02-25'),

('C118', 'Sam', '133 Pine St', '2024-06-01'),

('C119', 'John', '143 Main St', '2024-05-01');

SELECT \* FROM members;

-- Insert values into each branch table

INSERT INTO branch(branch\_id, manager\_id, branch\_address, contact\_no)

VALUES

('B001', 'E109', '123 Main St', '+919099988676'),

('B002', 'E109', '456 Elm St', '+919099988677'),

('B003', 'E109', '789 Oak St', '+919099988678'),

('B004', 'E110', '567 Pine St', '+919099988679'),

('B005', 'E110', '890 Maple St', '+919099988680');

SELECT \* FROM branch;

-- Insert values into each employees table

INSERT INTO employees(emp\_id, emp\_name, position, salary, branch\_id)

VALUES

('E101', 'John Doe', 'Clerk', 60000.00, 'B001'),

('E102', 'Jane Smith', 'Clerk', 45000.00, 'B002'),

('E103', 'Mike Johnson', 'Librarian', 55000.00, 'B001'),

('E104', 'Emily Davis', 'Assistant', 40000.00, 'B001'),

('E105', 'Sarah Brown', 'Assistant', 42000.00, 'B001'),

('E106', 'Michelle Ramirez', 'Assistant', 43000.00, 'B001'),

('E107', 'Michael Thompson', 'Clerk', 62000.00, 'B005'),

('E108', 'Jessica Taylor', 'Clerk', 46000.00, 'B004'),

('E109', 'Daniel Anderson', 'Manager', 57000.00, 'B003'),

('E110', 'Laura Martinez', 'Manager', 41000.00, 'B005'),

('E111', 'Christopher Lee', 'Assistant', 65000.00, 'B005');

SELECT \* FROM employees;

-- Inserting into books table

INSERT INTO books(isbn, book\_title, category, rental\_price, status, author, publisher)

VALUES

('978-0-553-29698-2', 'The Catcher in the Rye', 'Classic', 7.00, 'yes', 'J.D. Salinger', 'Little, Brown and Company'),

('978-0-330-25864-8', 'Animal Farm', 'Classic', 5.50, 'yes', 'George Orwell', 'Penguin Books'),

('978-0-14-118776-1', 'One Hundred Years of Solitude', 'Literary Fiction', 6.50, 'yes', 'Gabriel Garcia Marquez', 'Penguin Books'),

('978-0-525-47535-5', 'The Great Gatsby', 'Classic', 8.00, 'yes', 'F. Scott Fitzgerald', 'Scribner'),

('978-0-141-44171-6', 'Jane Eyre', 'Classic', 4.00, 'yes', 'Charlotte Bronte', 'Penguin Classics'),

('978-0-307-37840-1', 'The Alchemist', 'Fiction', 2.50, 'yes', 'Paulo Coelho', 'HarperOne'),

('978-0-679-76489-8', 'Harry Potter and the Sorcerers Stone', 'Fantasy', 7.00, 'yes', 'J.K. Rowling', 'Scholastic'),

('978-0-7432-4722-4', 'The Da Vinci Code', 'Mystery', 8.00, 'yes', 'Dan Brown', 'Doubleday'),

('978-0-09-957807-9', 'A Game of Thrones', 'Fantasy', 7.50, 'yes', 'George R.R. Martin', 'Bantam'),

('978-0-393-05081-8', 'A Peoples History of the United States', 'History', 9.00, 'yes', 'Howard Zinn', 'Harper Perennial'),

('978-0-19-280551-1', 'The Guns of August', 'History', 7.00, 'yes', 'Barbara W. Tuchman', 'Oxford University Press'),

('978-0-307-58837-1', 'Sapiens: A Brief History of Humankind', 'History', 8.00, 'no', 'Yuval Noah Harari', 'Harper Perennial'),

('978-0-375-41398-8', 'The Diary of a Young Girl', 'History', 6.50, 'no', 'Anne Frank', 'Bantam'),

('978-0-14-044930-3', 'The Histories', 'History', 5.50, 'yes', 'Herodotus', 'Penguin Classics'),

('978-0-393-91257-8', 'Guns, Germs, and Steel: The Fates of Human Societies', 'History', 7.00, 'yes', 'Jared Diamond', 'W. W. Norton & Company'),

('978-0-7432-7357-1', '1491: New Revelations of the Americas Before Columbus', 'History', 6.50, 'no', 'Charles C. Mann', 'Vintage Books'),

('978-0-679-64115-3', '1984', 'Dystopian', 6.50, 'yes', 'George Orwell', 'Penguin Books'),

('978-0-14-143951-8', 'Pride and Prejudice', 'Classic', 5.00, 'yes', 'Jane Austen', 'Penguin Classics'),

('978-0-452-28240-7', 'Brave New World', 'Dystopian', 6.50, 'yes', 'Aldous Huxley', 'Harper Perennial'),

('978-0-670-81302-4', 'The Road', 'Dystopian', 7.00, 'yes', 'Cormac McCarthy', 'Knopf'),

('978-0-385-33312-0', 'The Shining', 'Horror', 6.00, 'yes', 'Stephen King', 'Doubleday'),

('978-0-451-52993-5', 'Fahrenheit 451', 'Dystopian', 5.50, 'yes', 'Ray Bradbury', 'Ballantine Books'),

('978-0-345-39180-3', 'Dune', 'Science Fiction', 8.50, 'yes', 'Frank Herbert', 'Ace'),

('978-0-375-50167-0', 'The Road', 'Dystopian', 7.00, 'yes', 'Cormac McCarthy', 'Vintage'),

('978-0-06-025492-6', 'Where the Wild Things Are', 'Children', 3.50, 'yes', 'Maurice Sendak', 'HarperCollins'),

('978-0-06-112241-5', 'The Kite Runner', 'Fiction', 5.50, 'yes', 'Khaled Hosseini', 'Riverhead Books'),

('978-0-06-440055-8', 'Charlotte''s Web', 'Children', 4.00, 'yes', 'E.B. White', 'Harper & Row'),

('978-0-679-77644-3', 'Beloved', 'Fiction', 6.50, 'yes', 'Toni Morrison', 'Knopf'),

('978-0-14-027526-3', 'A Tale of Two Cities', 'Classic', 4.50, 'yes', 'Charles Dickens', 'Penguin Books'),

('978-0-7434-7679-3', 'The Stand', 'Horror', 7.00, 'yes', 'Stephen King', 'Doubleday'),

('978-0-451-52994-2', 'Moby Dick', 'Classic', 6.50, 'yes', 'Herman Melville', 'Penguin Books'),

('978-0-06-112008-4', 'To Kill a Mockingbird', 'Classic', 5.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.'),

('978-0-553-57340-1', '1984', 'Dystopian', 6.50, 'yes', 'George Orwell', 'Penguin Books'),

('978-0-7432-4722-5', 'Angels & Demons', 'Mystery', 7.50, 'yes', 'Dan Brown', 'Doubleday'),

('978-0-7432-7356-4', 'The Hobbit', 'Fantasy', 7.00, 'yes', 'J.R.R. Tolkien', 'Houghton Mifflin Harcourt');

-- inserting into issued table

INSERT INTO issued\_status(issued\_id, issued\_member\_id, issued\_book\_name, issued\_date, issued\_book\_isbn, issued\_emp\_id)

VALUES

('IS106', 'C106', 'Animal Farm', '2024-03-10', '978-0-330-25864-8', 'E104'),

('IS107', 'C107', 'One Hundred Years of Solitude', '2024-03-11', '978-0-14-118776-1', 'E104'),

('IS108', 'C108', 'The Great Gatsby', '2024-03-12', '978-0-525-47535-5', 'E104'),

('IS109', 'C109', 'Jane Eyre', '2024-03-13', '978-0-141-44171-6', 'E105'),

('IS110', 'C110', 'The Alchemist', '2024-03-14', '978-0-307-37840-1', 'E105'),

('IS111', 'C109', 'Harry Potter and the Sorcerers Stone', '2024-03-15', '978-0-679-76489-8', 'E105'),

('IS112', 'C109', 'A Game of Thrones', '2024-03-16', '978-0-09-957807-9', 'E106'),

('IS113', 'C109', 'A Peoples History of the United States', '2024-03-17', '978-0-393-05081-8', 'E106'),

('IS114', 'C109', 'The Guns of August', '2024-03-18', '978-0-19-280551-1', 'E106'),

('IS115', 'C109', 'The Histories', '2024-03-19', '978-0-14-044930-3', 'E107'),

('IS116', 'C110', 'Guns, Germs, and Steel: The Fates of Human Societies', '2024-03-20', '978-0-393-91257-8', 'E107'),

('IS117', 'C110', '1984', '2024-03-21', '978-0-679-64115-3', 'E107'),

('IS118', 'C101', 'Pride and Prejudice', '2024-03-22', '978-0-14-143951-8', 'E108'),

('IS119', 'C110', 'Brave New World', '2024-03-23', '978-0-452-28240-7', 'E108'),

('IS120', 'C110', 'The Road', '2024-03-24', '978-0-670-81302-4', 'E108'),

('IS121', 'C102', 'The Shining', '2024-03-25', '978-0-385-33312-0', 'E109'),

('IS122', 'C102', 'Fahrenheit 451', '2024-03-26', '978-0-451-52993-5', 'E109'),

('IS123', 'C103', 'Dune', '2024-03-27', '978-0-345-39180-3', 'E109'),

('IS124', 'C104', 'Where the Wild Things Are', '2024-03-28', '978-0-06-025492-6', 'E110'),

('IS125', 'C105', 'The Kite Runner', '2024-03-29', '978-0-06-112241-5', 'E110'),

('IS126', 'C105', 'Charlotte''s Web', '2024-03-30', '978-0-06-440055-8', 'E110'),

('IS127', 'C105', 'Beloved', '2024-03-31', '978-0-679-77644-3', 'E110'),

('IS128', 'C105', 'A Tale of Two Cities', '2024-04-01', '978-0-14-027526-3', 'E110'),

('IS129', 'C105', 'The Stand', '2024-04-02', '978-0-7434-7679-3', 'E110'),

('IS130', 'C106', 'Moby Dick', '2024-04-03', '978-0-451-52994-2', 'E101'),

('IS131', 'C106', 'To Kill a Mockingbird', '2024-04-04', '978-0-06-112008-4', 'E101'),

('IS132', 'C106', 'The Hobbit', '2024-04-05', '978-0-7432-7356-4', 'E106'),

('IS133', 'C107', 'Angels & Demons', '2024-04-06', '978-0-7432-4722-5', 'E106'),

('IS134', 'C107', 'The Diary of a Young Girl', '2024-04-07', '978-0-375-41398-8', 'E106'),

('IS135', 'C107', 'Sapiens: A Brief History of Humankind', '2024-04-08', '978-0-307-58837-1', 'E108'),

('IS136', 'C107', '1491: New Revelations of the Americas Before Columbus', '2024-04-09', '978-0-7432-7357-1', 'E102'),

('IS137', 'C107', 'The Catcher in the Rye', '2024-04-10', '978-0-553-29698-2', 'E103'),

('IS138', 'C108', 'The Great Gatsby', '2024-04-11', '978-0-525-47535-5', 'E104'),

('IS139', 'C109', 'Harry Potter and the Sorcerers Stone', '2024-04-12', '978-0-679-76489-8', 'E105'),

('IS140', 'C110', 'Animal Farm', '2024-04-13', '978-0-330-25864-8', 'E102'),

('IS151', 'C118', 'The Catcher in the Rye', SUBDATE(CURRENT\_DATE, INTERVAL 24 DAY), '978-0-553-29698-2', 'E108'),

('IS152', 'C119', 'The Catcher in the Rye', SUBDATE(CURRENT\_DATE, INTERVAL 13 DAY), '978-0-553-29698-2', 'E109'),

('IS153', 'C106', 'Pride and Prejudice', SUBDATE(CURRENT\_DATE, INTERVAL 7 DAY), '978-0-14-143951-8', 'E107'),

('IS154', 'C105', 'The Road', SUBDATE(CURRENT\_DATE, INTERVAL 32 DAY), '978-0-375-50167-0', 'E101');

SELECT \* FROM issued\_status;

-- inserting into return table

INSERT INTO return\_status(return\_id, issued\_id, return\_date)

VALUES

('RS101', 'IS101', '2023-06-06'),

('RS102', 'IS105', '2023-06-07'),

('RS103', 'IS103', '2023-08-07'),

('RS104', 'IS106', '2024-05-01'),

('RS105', 'IS107', '2024-05-03'),

('RS106', 'IS108', '2024-05-05'),

('RS107', 'IS109', '2024-05-07'),

('RS108', 'IS110', '2024-05-09'),

('RS109', 'IS111', '2024-05-11'),

('RS110', 'IS112', '2024-05-13'),

('RS111', 'IS113', '2024-05-15'),

('RS112', 'IS114', '2024-05-17'),

('RS113', 'IS115', '2024-05-19'),

('RS114', 'IS116', '2024-05-21'),

('RS115', 'IS117', '2024-05-23'),

('RS116', 'IS118', '2024-05-25'),

('RS117', 'IS119', '2024-05-27'),

('RS118', 'IS120', '2024-05-29');

ALTER TABLE return\_status

ADD Column book\_quality VARCHAR(15) DEFAULT('Good');

SET sql\_safe\_updates=0;

UPDATE return\_status

SET book\_quality = 'Damaged'

WHERE issued\_id IN ('IS112', 'IS117', 'IS118');

SELECT \* FROM return\_status;

-- Query 1. Create a New Book Record

-- "978-1-60129-456-2', 'To Kill a Mockingbird', 'Classic', 6.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.')"

insert into books values("978-1-60129-456-2", 'To Kill a Mockingbird',

'Classic', 6.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.');

select \* from books;

--Expalnation

-- The query inserts a new book record for "To Kill a Mockingbird" into the books table with details like ISBN, genre, price, stock status, author, and publisher.

-- Then, it retrieves all records from the books table to display the updated list.

-- Query2: Update an Existing Member's Address

update members set member\_address="125 Main St" where member\_id="C101";

select \* from members;

--Explanation

-- The query updates the member\_address of the member with member\_id "C101" to "125 Main St" in the members table.

-- It then retrieves all records from the members table to display the updated information.

-- Query 3: Delete a Record from the Issued Status Table

-- Objective: Delete the record with issued\_id = 'IS107' from the issued\_status table.

delete from issued\_status where issued\_id="IS107";

--Explanation

-- The query deletes the record from the issued\_status table where issued\_id is "IS107".

-- This operation removes the specified entry, and any related data associated with this issued\_id will no longer exist in the table.

-- Query 4: Retrieve All Books Issued by a Specific Employee

-- Objective: Select all books issued by the employee with emp\_id = 'E101'.

select \* from issued\_status where issued\_emp\_id="E101"

--Explanation

-- The query retrieves all columns and records from the `issued\_status` table where the `issued\_emp\_id` is "E101".

-- It displays the complete information for all entries related to the employee with the ID "E101".

-- Query 5: List Members Who Have Issued More Than One Book

-- Objective: Use GROUP BY to find members who have issued more than one book.

select issued\_member\_id from issued\_status

group by issued\_member\_id

having count(issued\_member\_id)>1;

--Explanation

-- The query selects the `issued\_member\_id` from the `issued\_status` table and groups the results by `issued\_member\_id`.

-- It then filters these groups using the `HAVING` clause to show only those `issued\_member\_id`s that appear \*\*more than once\*\* (i.e., having a count greater than 1).

-- This helps identify members who have multiple issued records.

-- Query 6: Create Summary Tables\*\*: Used CTAS to generate new tables based on query results - each book and total book\_issued\_cnt

create table book\_cnts as

select

b.isbn,b.book\_title,

count(ist.issued\_id) as no\_issued

from books as b

join

issued\_status as ist

on ist.issued\_book\_isbn=b.isbn

group by b.isbn,b.book\_title;

select \* from book\_cnts;

--Explanation

-- The query creates a new table named book\_cnts that contains three columns: isbn, book\_title, and no\_issued.

-- It joins the books table with the issued\_status table based on the matching ISBN (issued\_book\_isbn). For each book, it counts the number of times it has been issued (issued\_id) and stores this count as no\_issued.

-- the SELECT statement retrieves all records from book\_cnts, showing the ISBN, title, and total issue count for each book.

-- Query 7. \*\*Retrieve All Books in a Specific Category:

select \* from books

where category="classic";

--Explanation

-- This query will return all book records from the books table that belong to the "classic" genre, displaying details such as ISBN, title, author, price, etc., for each classic book.

-- Query 8: Find Total Rental Income by Category:

select

b.category, sum(b.rental\_price) as total\_rent\_price

from books as b

join

issued\_status as ist

on ist.issued\_book\_isbn=b.isbn

group by b.category;

--Explanation

-- The query joins the books table with the issued\_status table on the isbn and issued\_book\_isbn columns to filter out only the books that have been issued.

-- It then calculates the sum of the rental\_price for each book category and groups the results by the category of the book.

-- Query 9. \*\*List Members Who Registered in the Last 180 Days\*\*:

select \* from members

where reg\_date>=subdate(current\_date(),interval 180 day);

--Explanation

-- The query selects all records from the members table where the reg\_date (registration date) is within the last 180 days from the current date.

-- Query 10: List Employees with Their Branch Manager's Name and their branch details\*\*:

select e1.\*,e2.emp\_name as manager,b.manager\_id from

employees as e1

join

branch as b

on b.branch\_id=e1.branch\_id

join employees as e2

on b.manager\_id =e2.emp\_id;

--Explanation

-- The query returns all employee details (e1.\*), their manager's name (manager), and the manager\_id from the branch table.

-- It’s useful for identifying employees along with their respective managers based on branch data.

-- Query 11. Create a Table of Books with Rental Price Above a Certain Threshold 7USD:

create table books\_price\_greater\_than\_seven as

select \* from books

where rental\_price>7;

select \* from books\_price\_greater\_than\_seven;

--Explanation

-- The first query creates a new table `books\_price\_greater\_than\_seven` with books whose rental price is greater than 7.

-- The second query selects and displays all records from books\_price\_greater\_than\_seven

-- Query 12: Retrieve the List of Books Not Yet Returned

select ist.issued\_book\_name

from issued\_status as ist

left join

return\_status as rs

on ist.issued\_id=rs.issued\_id

where rs.return\_id is null;

--Exlanation

-- The query selects the names of books from the issued\_status table that have not been returned yet by performing a LEFT JOIN with the return\_status table.

-- It filters the results where the return\_id is NULL, indicating the books are still issued.

-- Query 13: Identify Members with Overdue Books

-- Write a query to identify members who have overdue books (assume a 30-day return period). Display the member's name, book title, issue date, and days overdue.

SELECT

ist.issued\_member\_id,

m.member\_name,

bk.book\_title,

ist.issued\_date,

-- rs.return\_date,

CURRENT\_DATE - ist.issued\_date as over\_dues\_days

FROM issued\_status as ist

JOIN

members as m

ON m.member\_id = ist.issued\_member\_id

JOIN

books as bk

ON bk.isbn = ist.issued\_book\_isbn

LEFT JOIN

return\_status as rs

ON rs.issued\_id = ist.issued\_id

WHERE

rs.return\_date IS NULL

AND

(CURRENT\_DATE - ist.issued\_date) > 30

ORDER BY ist.issued\_member\_id;

--Explanation

-- This query is designed to retrieve information about overdue books that have been issued for more than 30 days and have not been returned.

-- It selects the member ID (issued\_member\_id) and name (member\_name), the title of the book (book\_title), the date the book was issued (issued\_date), and calculates the number of overdue days (over\_dues\_days) by subtracting the issued\_date from the current date.

-- The data is sourced from the issued\_status table, with joins to the members and books tables to fetch the relevant details about the member and the book.

-- A LEFT JOIN with the return\_status table ensures that only books that have not been returned are included, as it checks where return\_date is NULL.

-- The query filters for books that have been issued for more than 30 days and sorts the results by issued\_member\_id to group the books by the member.

-- Query 14: Update Book Status on Return

-- Write a query to update the status of books in the books table to "available" when they are returned (based on entries in the return\_status table).

-- Store Procedures

delimiter $$

CREATE PROCEDURE add\_return\_records(in p\_return\_id VARCHAR(10),in p\_issued\_id VARCHAR(10),in p\_book\_quality VARCHAR(10))

Begin

DECLARE v\_isbn VARCHAR(50);

DECLARE v\_book\_name VARCHAR(80);

INSERT INTO return\_status(return\_id, issued\_id, return\_date, book\_quality)

VALUES

(p\_return\_id, p\_issued\_id, CURRENT\_DATE, p\_book\_quality);

SELECT

issued\_book\_isbn,

issued\_book\_name

INTO

v\_isbn,

v\_book\_name

FROM issued\_status

WHERE issued\_id = p\_issued\_id;

UPDATE books

SET status = 'yes'

WHERE isbn = v\_isbn;

select concat('Thank you for returning the book: %', v\_book\_name) AS message;

END $$

delimiter ;

CALL add\_return\_records('RS138','IS135','GOOD');

CALL add\_return\_records('RS148', 'IS140', 'Good');

--Expalnation

-- The add\_return\_records procedure manages the process of recording a book return in a library system.

-- It takes three input parameters: p\_return\_id (the ID for the return record), p\_issued\_id (the ID of the issued book), and p\_book\_quality (the condition of the returned book). First, the procedure inserts a new record into the return\_status table with the provided details, including the current date as the return date and the specified book quality.

-- It then fetches the isbn and book\_name from the issued\_status table based on the given p\_issued\_id, storing them in variables.

-- The procedure proceeds by updating the books table, setting the status of the book (identified by the ISBN) to 'yes', marking it as returned.

-- It constructs and displays a message that thanks the user for returning the book, using the book's name in the message.

-- This stored procedure ensures that return records are properly logged, the book's status is updated, and the user receives acknowledgment for their return.

-- Query 15: Branch Performance Report

-- Create a query that generates a performance report for each branch, showing the number of books issued, the number of books returned, and the total revenue generated from book rentals.

CREATE TABLE branch\_reports

AS

SELECT

b.branch\_id,

b.manager\_id,

COUNT(ist.issued\_id) as number\_book\_issued,

COUNT(rs.return\_id) as number\_of\_book\_return,

SUM(bk.rental\_price) as total\_revenue

FROM issued\_status as ist

JOIN

employees as e

ON e.emp\_id = ist.issued\_emp\_id

JOIN

branch as b

ON e.branch\_id = b.branch\_id

LEFT JOIN

return\_status as rs

ON rs.issued\_id = ist.issued\_id

JOIN

books as bk

ON ist.issued\_book\_isbn = bk.isbn

GROUP BY b.branch\_id, b.manager\_id

order by b.branch\_id, b.manager\_id;

SELECT \* FROM branch\_reports;

--Explanation

-- The branch\_reports table is created by aggregating data from multiple tables, including issued\_status, return\_status, books, employees, and branch.

-- It calculates the total number of books issued, the number of books returned, and the total revenue for each branch, grouped by branch\_id and manager\_id.

-- The result provides insights into the activity and financial performance of each branch, such as the number of transactions and the revenue generated from book rentals, with the data sorted by branch and manager for easy analysis.

-- Query 16: CTAS: Create a Table of Active Members

-- Use the CREATE TABLE AS (CTAS) statement to create a new table active\_members containing members who have issued at least one book in the last 2 months.

CREATE TABLE active\_members

AS

SELECT \* FROM members

WHERE member\_id IN (SELECT

DISTINCT issued\_member\_id

FROM issued\_status

WHERE

issued\_date >= subdate(CURDATE(),INTERVAL 2 month)

);

SELECT \* FROM active\_members;

--Explanation

-- The active\_members table is designed to identify and store members who have actively issued books within the last two months.

-- It first selects distinct member IDs from the issued\_status table where the issued\_date is within the last two months, ensuring only members who have recently borrowed books are included.

-- The outer query then pulls the full details of these members from the members table by matching the member\_id to those found in the subquery.

-- This process creates a list of active members, offering a way to track recent book borrowers

-- Query 17: Find Employees with the Most Book Issues Processed

-- Write a query to find the top 3 employees who have processed the most book issues. Display the employee name, number of books processed, and their branch.

SELECT

e.emp\_name,

b.\*,

COUNT(ist.issued\_id) as no\_book\_issued

FROM issued\_status as ist

JOIN

employees as e

ON e.emp\_id = ist.issued\_emp\_id

JOIN

branch as b

ON e.branch\_id = b.branch\_id

GROUP BY 1, 2;

--Explanation

-- This query retrieves the name of each employee, their associated branch details, and the total number of books they have issued, grouped by employee and branch.

-- The COUNT(ist.issued\_id) provides a count of issued books per employee, helping to track employee activity and performance.

-- The GROUP BY ensures the results are aggregated by employee and branch, allowing insights into book issuance at the branch level.

-- Query 18: Stored Procedure

-- Objective: Create a stored procedure to manage the status of books in a library system.

/\* Description: Write a stored procedure that updates the status of a book based on its issuance or return. Specifically:

If a book is issued, the status should change to 'no'.

If a book is returned, the status should change to 'yes'.

\*/

delimiter $$

CREATE PROCEDURE issue\_book(p\_issued\_id VARCHAR(10), p\_issued\_member\_id VARCHAR(30), p\_issued\_book\_isbn VARCHAR(30), p\_issued\_emp\_id VARCHAR(10))

BEGIN

DECLARE v\_status VARCHAR(10);

SELECT

status

INTO

v\_status

FROM books

WHERE isbn = p\_issued\_book\_isbn;

IF v\_status = 'yes' THEN

INSERT INTO issued\_status(issued\_id, issued\_member\_id, issued\_date, issued\_book\_isbn, issued\_emp\_id)

VALUES

(p\_issued\_id, p\_issued\_member\_id, CURRENT\_DATE, p\_issued\_book\_isbn, p\_issued\_emp\_id);

UPDATE books

SET status = 'no'

WHERE isbn = p\_issued\_book\_isbn;

select concat('Book records added successfully for book isbn : %', p\_issued\_book\_isbn);

ELSE

select concat('Sorry to inform you the book you have requested is unavailable book\_isbn: %', p\_issued\_book\_isbn);

END IF;

END $$

delimeter ;

CALL issue\_book('IS155', 'C108', '978-0-553-29698-2', 'E104');

CALL issue\_book('IS156', 'C108', '978-0-375-41398-8', 'E104');

--Explanation

-- The issue\_book stored procedure is designed to handle the issuance of books in a library system.

-- It accepts four input parameters: the issue ID, member ID, book ISBN, and employee ID. The procedure first checks if the book is available by querying its status from the books table.

-- If the book's status is 'yes' (available), it inserts a new record into the issued\_status table with the provided details (issue ID, member ID, book ISBN, and employee ID), updates the book's status to 'no' (marking it as issued), and displays a success message confirming the issuance.

-- If the book is unavailable (status is not 'yes'), the procedure displays a message informing the user that the book is not available. The procedure ensures that only available books are issued and provides immediate feedback based on the book's availability.