LIBRARY MANAGEMENT SYSTEM SQL ANALYSIS

Data Modeling

ALTER TABLE issued\_status

ADD CONSTRAINT fk\_members

FOREIGN KEY (issued\_member\_id)

REFERENCES members (member\_id);

ALTER TABLE issued\_status

ADD CONSTRAINT fk\_books

FOREIGN KEY (issued\_book\_isbn)

REFERENCES books (isbn);

ALTER TABLE issued\_status

ADD CONSTRAINT fk\_employees

FOREIGN KEY (issued\_emp\_id)

REFERENCES employees (emp\_id);

ALTER TABLE employees

ADD CONSTRAINT fk\_branch

FOREIGN KEY (branch\_id)

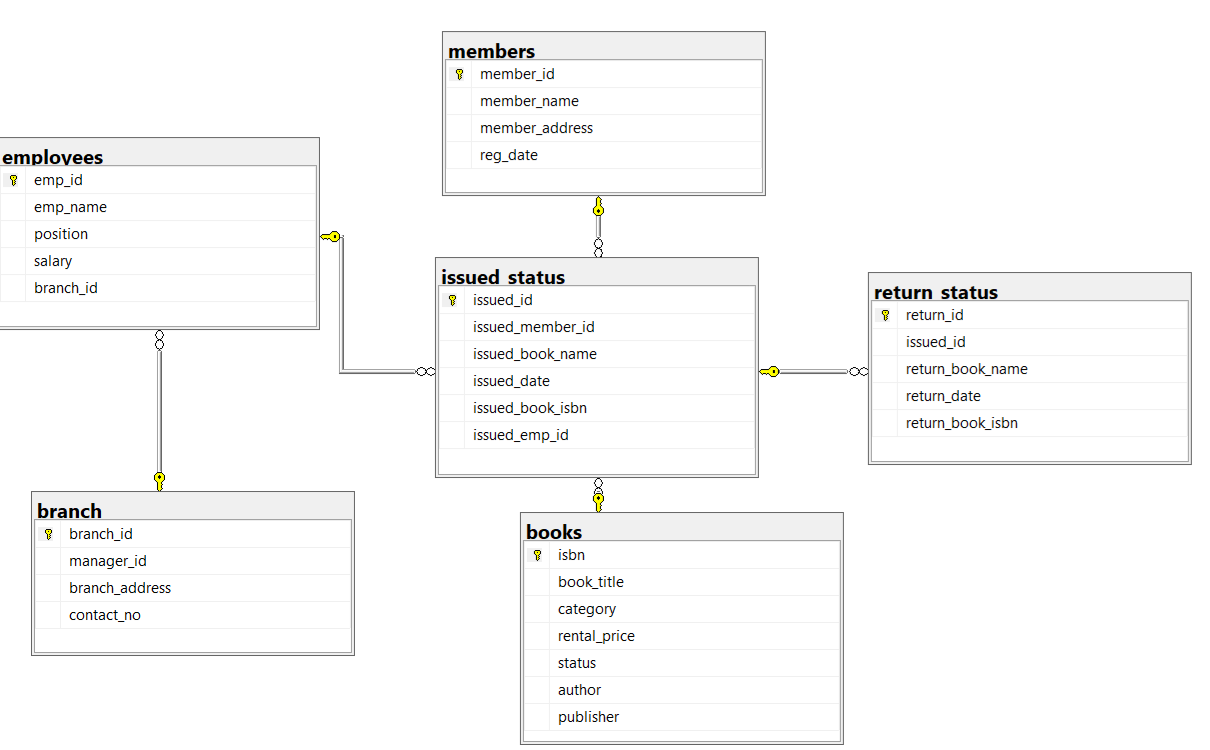
REFERENCES branch (branch\_id);

ALTER TABLE return\_status

ADD CONSTRAINT fk\_issued\_status

FOREIGN KEY (issued\_id)

REFERENCES issued\_status(issued\_id);



-- While solving above query I got this error

The ALTER TABLE statement conflicted with the FOREIGN KEY constraint "fk\_issued\_status". The conflict occurred in database "Library\_Project\_DB", table "dbo.issued\_status", column 'issued\_id'

Then We found there are return\_status.issued\_id are not present in issued\_status.issued\_id so I removed it by Delete query as done below

SELECT issued\_id

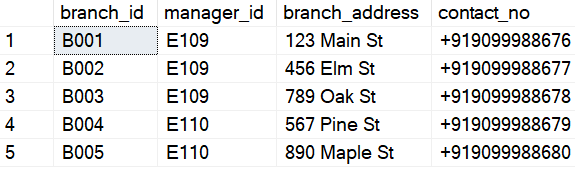
FROM return\_status

WHERE issued\_id NOT IN (SELECT issued\_id FROM issued\_status);

DELETE FROM return\_status

WHERE issued\_id NOT IN (SELECT issued\_id FROM issued\_status);

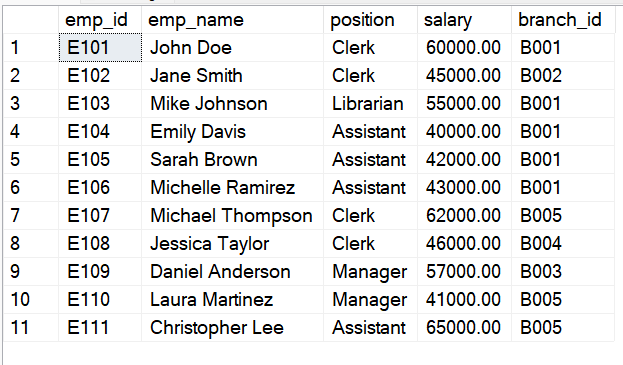
SELECT \* FROM branch



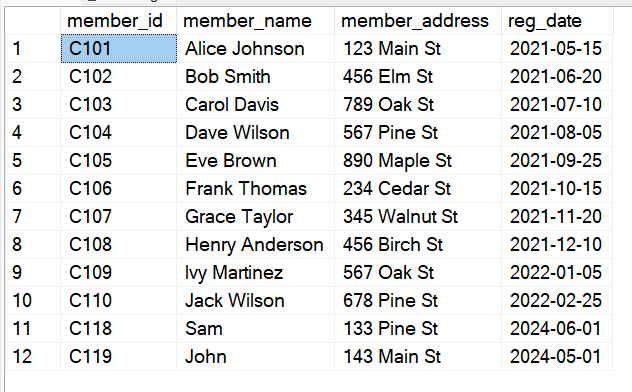
SELECT \* FROM books



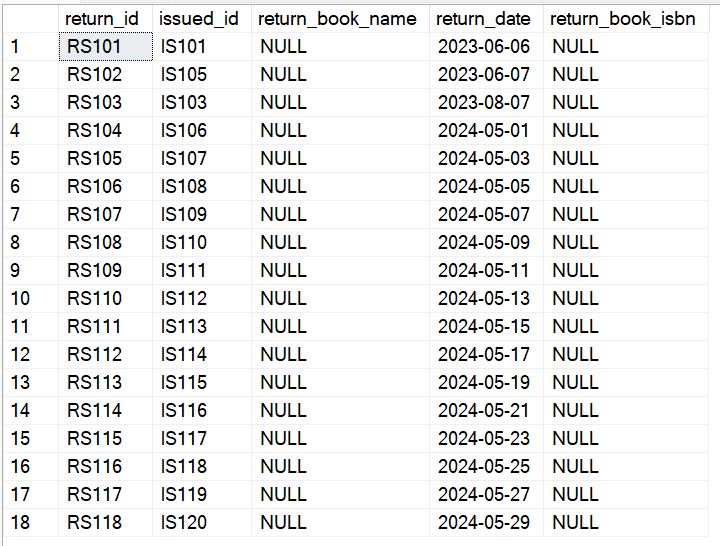
SELECT \* FROM employees



SELECT \* FROM members



SELECT \* FROM return\_status;



SELECT \* FROM issued\_status



**Project TASK**

**CRUD Operations**

Task 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.')"

SELECT TOP (1000) [isbn]

,[book\_title]

,[category]

,[rental\_price]

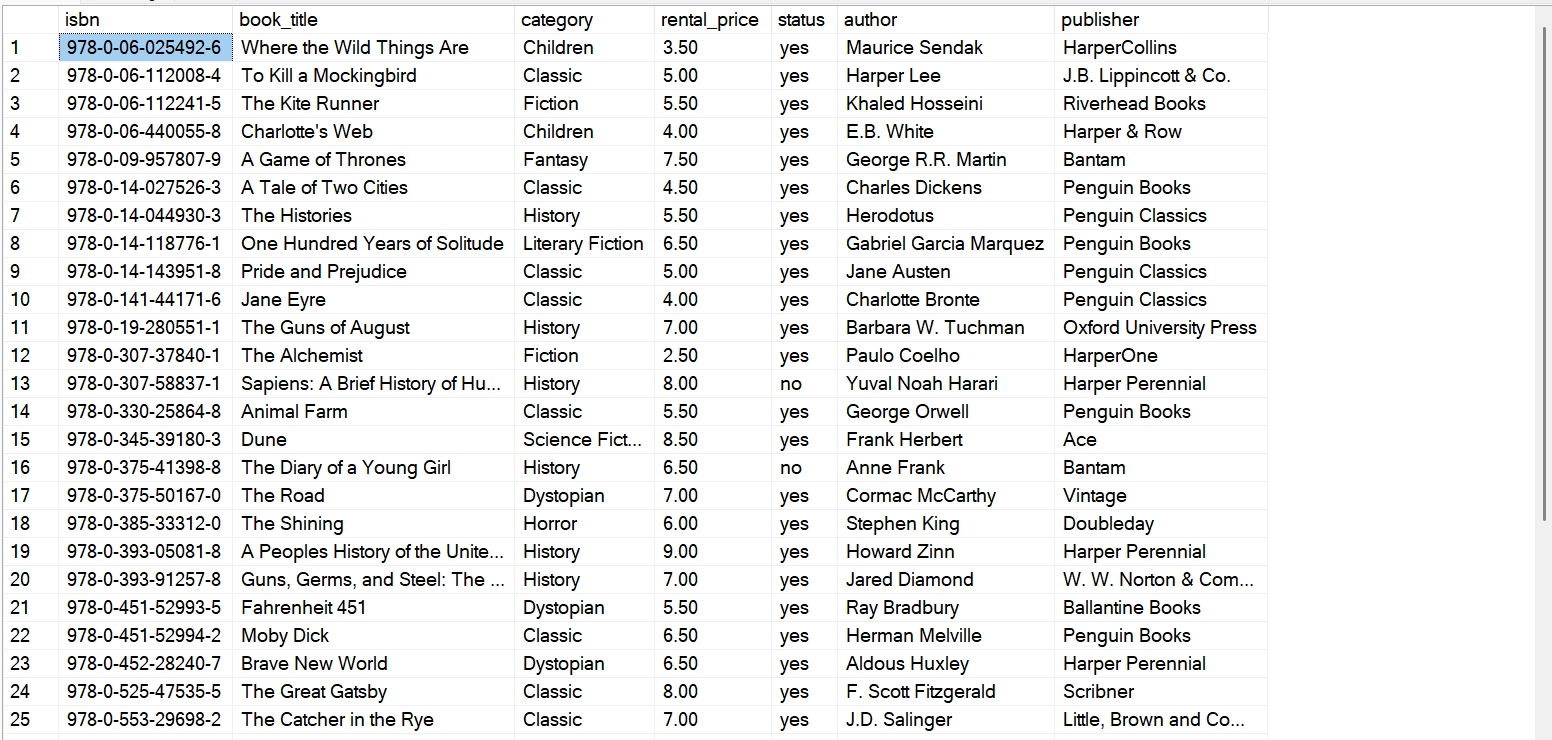
,[status]

,[author]

,[publisher]

FROM [Library\_Project\_DB].[dbo].[books]

-- Before we have 35 records



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

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



Task 2: Update an Existing Member's Address

UPDATE members

SET member\_address = '125 Main st'

Where member\_id= 'C101'

SELECT \* FROM members



Task 3: Delete a Record from the Issued Status Table

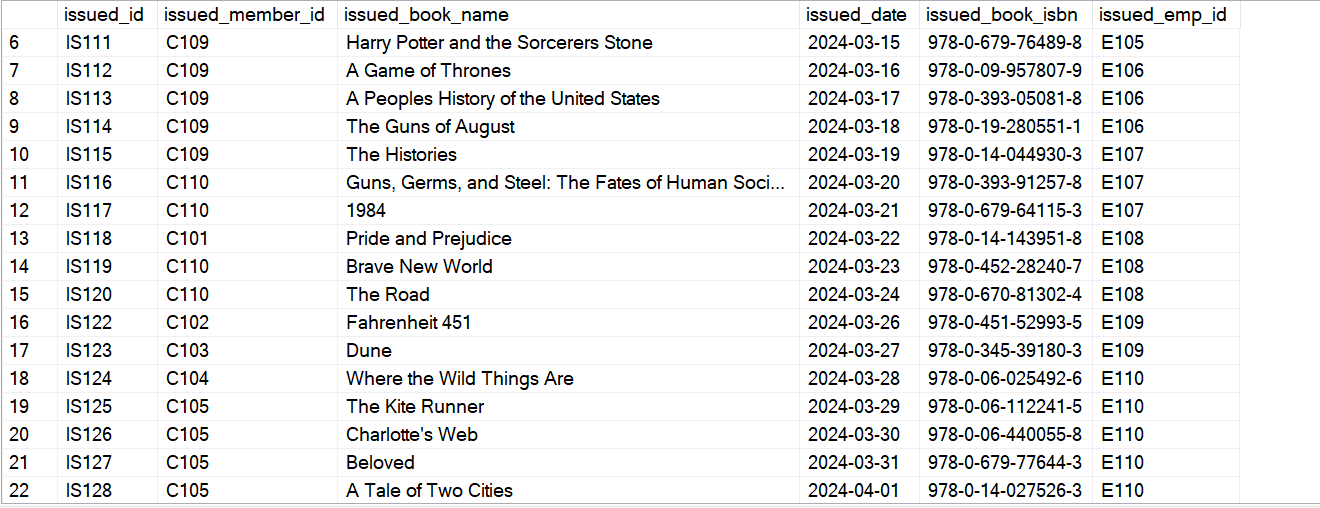
Objective: Delete the record with issued\_id = 'IS104' from the issued\_status table.

Select \* from issued\_status-- There were 35 rows

DELETE FROM issued\_status

WHERE issued\_id = 'IS121';

Select \* from issued\_status-- Now there are 34 Rows

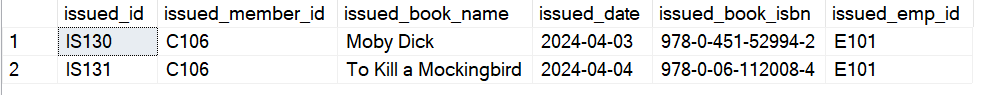


Task 4: Retrieve All Books Issued by a Specific Employee

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

SELECT \* FROM issued\_status

WHERE issued\_emp\_id = 'E101'



Task 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 \* FROM

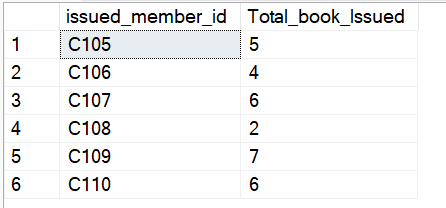
(

SELECT issued\_member\_id, COUNT(\*) AS Total\_book\_Issued FROM issued\_status

Group by issued\_member\_id

)t

WHERE Total\_book\_Issued >1



**CTAS (Create Table As Select)**

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

SELECT

b.isbn,

b.book\_title,

COUNT(isst.issued\_id) AS Total\_book\_Issued

INTO Total\_Books\_Issued\_by\_Each\_book

FROM books as b

JOIN issued\_status as isst

ON isst.issued\_book\_isbn = b.isbn

Group by b.isbn, b.book\_title

Select \* from Total\_Books\_Issued\_by\_Each\_book



**Data Analysis & Findings**

Task 7. Retrieve All Books in a Specific Category:

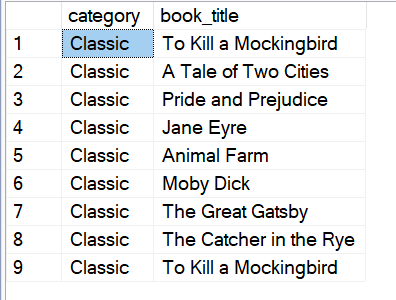
Objective: Category is Classic

SELECT

category,

book\_title FROM books

WHERE category = 'Classic'



Task 8: Find Total Rental Income by Category:

SELECT

b.category,

SUM(b.rental\_price) AS Rental\_Income,

COUNT(\*) No\_of\_items\_sold

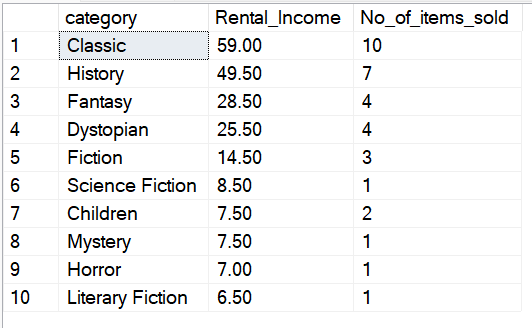
FROM books AS b

JOIN issued\_status as isst

ON isst.issued\_book\_isbn = b.isbn

GROUP BY category

ORDER BY Rental\_Income DESC

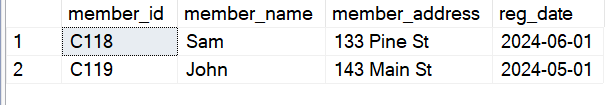


Task 9. List Members Who Registered in the Last 180 Days:

SELECT \*

FROM members

WHERE reg\_date > DATEADD(DAY, -300, GETDATE());



Task 10: List Employees with Their Branch Manager's Name and their branch details:

SELECT

e1.\*,

br.manager\_id,

e2.emp\_name as Manager

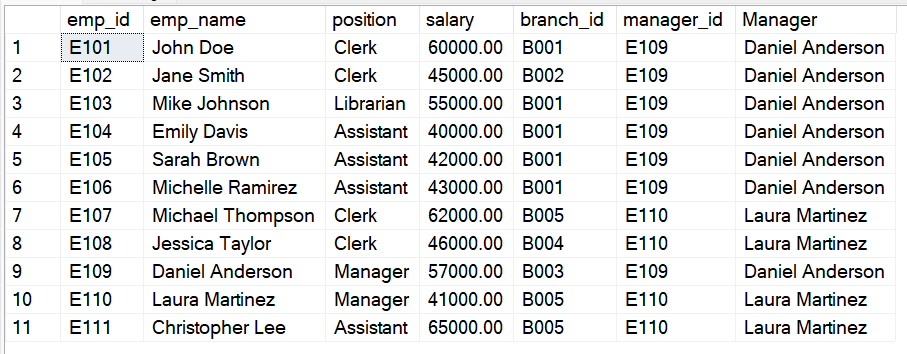
FROM employees as e1

JOIN branch br

ON e1.branch\_id = br.branch\_id

JOIN employees e2

ON e2.emp\_id = br.manager\_id



Task 11. Create a Table of Books with Rental Price Above a Certain Threshold

SELECT \* INTO Books\_Price\_Greater\_than\_Seven

FROM books

WHERE rental\_price > 7

SELECT \* FROM Books\_Price\_Greater\_than\_Seven



Task 12: Retrieve the List of Books Not Yet Returned

SELECT

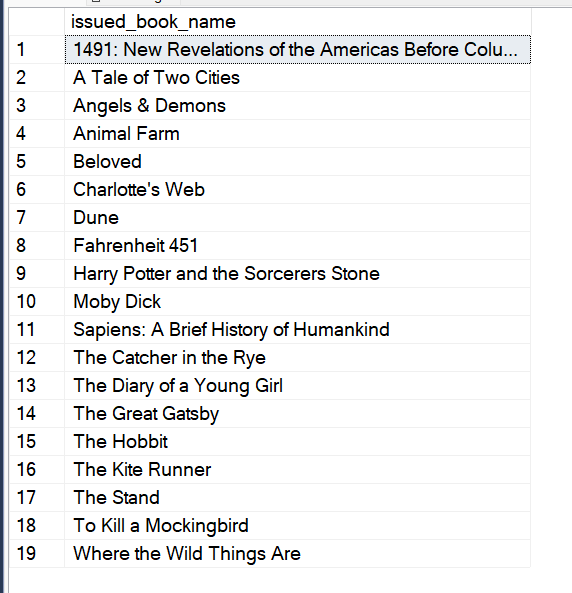
DISTINCT isst.issued\_book\_name

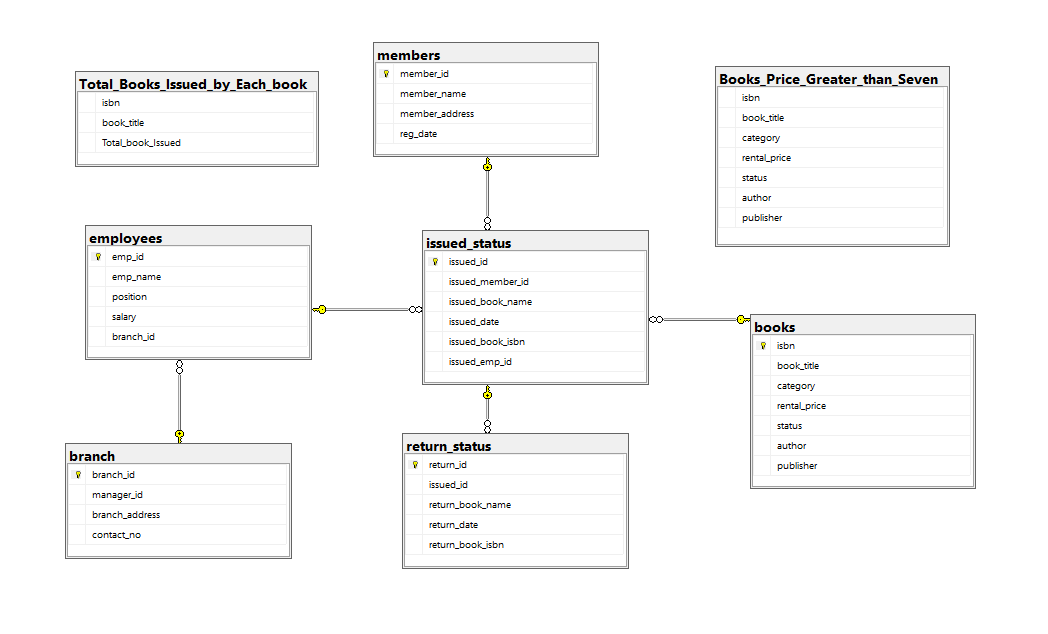
FROM issued\_status isst

LEFT JOIN [return\_status ] rs

ON isst.issued\_id = rs.issued\_id

WHERE rs.return\_id IS NULL





**SOME CHANGES BEFORE DOING ADVANCE SQL OPERATIONS**

-- Inserting records into issued\_status table

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

VALUES

('IS151', 'C118', 'The Catcher in the Rye', DATEADD(DAY, -144, GETDATE()), '978-0-553-29698-2', 'E108'),

('IS152', 'C119', 'The Catcher in the Rye', DATEADD(DAY, -133, GETDATE()), '978-0-553-29698-2', 'E109'),

('IS153', 'C106', 'Pride and Prejudice', DATEADD(DAY, -127, GETDATE()), '978-0-14-143951-8', 'E107'),

('IS154', 'C105', 'The Road', DATEADD(DAY, -152, GETDATE()), '978-0-375-50167-0', 'E101');

SELECT \* FROM issued\_status



-- Adding a new column to return\_status table

ALTER TABLE return\_status

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

-- Updating the book\_quality column for specific issued IDs

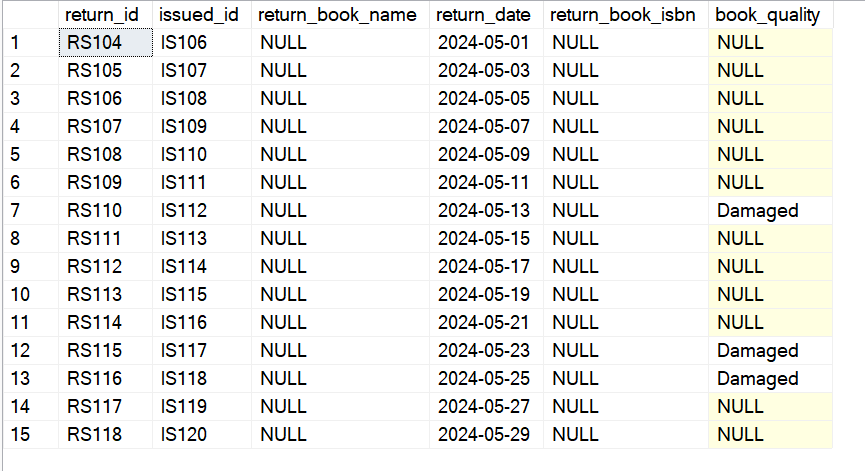
UPDATE [return\_status ]

SET book\_quality = 'Damaged'

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

-- Selecting all records from return\_status table

SELECT \* FROM [return\_status ];



ADVANCE SQL OPERATIONS

SELECT \* FROM books --36 Records

SELECT \* FROM branch -- 5 Records

SELECT \* FROM employees -- 11 Records

SELECT \* FROM issued\_status --38 Records

SELECT \* FROM members -- 12 Records

SELECT \* FROM [return\_status ] -- 15 Records

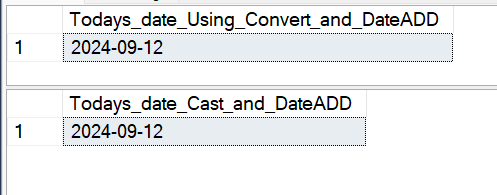
Task 13: Identify Members with Overdue Books

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 CONVERT(date, DATEADD(MONTH, -4, GETDATE())) Todays\_date\_Using\_Convert\_and\_DateADD;

-- OR

SELECT CAST(DATEADD(MONTH,-4,GETDATE()) AS date) Todays\_date\_Cast\_and\_DateADD



SELECT

isst.issued\_member\_id,

m.member\_name,

b.book\_title,

isst.issued\_date,

--rs.return\_date,

CAST(DATEADD(MONTH,-4,GETDATE())AS date) Todays\_date,

DATEDIFF(DAY,isst.issued\_date,

CAST(DATEADD(MONTH,-4,GETDATE()) AS date)) as Over\_due\_days

FROM issued\_status isst

JOIN members m

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

JOIN books b

ON isst.issued\_book\_isbn = b.isbn

LEFT JOIN [return\_status ] rs

ON isst.issued\_id = rs.issued\_id

WHERE

rs.return\_date is NULL

AND

DATEDIFF(DAY,isst.issued\_date,CAST(DATEADD(MONTH,-4,GETDATE()) AS date)) >30

ORDER BY 1



Task 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

GO

-- Stored Procedure for SQL Server

CREATE PROCEDURE add\_return\_records

@p\_return\_id VARCHAR(10),

@p\_issued\_id VARCHAR(10),

@p\_book\_quality VARCHAR(15)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @v\_isbn VARCHAR(50);

DECLARE @v\_book\_name VARCHAR(100);

-- Inserting Records in return\_status table based on user input

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

VALUES

(@p\_return\_id, @p\_issued\_id, CAST(GETDATE() AS DATE), @p\_book\_quality);

-- Retrieving book details from issued\_status

SELECT

@v\_isbn = issued\_book\_isbn,

@v\_book\_name = issued\_book\_name

FROM

issued\_status

WHERE

issued\_id = @p\_issued\_id;

-- Updating the book status in the books table

UPDATE books

SET status = 'yes'

WHERE isbn = @v\_isbn;

-- Output message for successful book return

PRINT 'Thank you for returning the book ' + @v\_book\_name;

END;

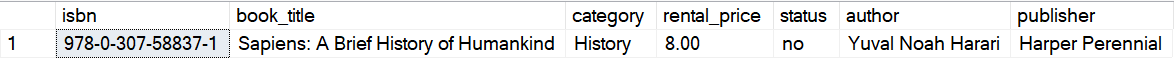
SELECT \* FROM issued\_status

SELECT \* FROM [return\_status ]

-- Below is the Verification that this isbn 978-0-307-58837-1 book is issued by IS135 issue\_id which has 'no' status as It is not available

SELECT \* FROM books

WHERE isbn ='978-0-307-58837-1'

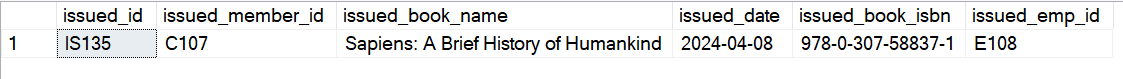


-- Here you can see book is not returned yet

-- By whom this book is issued

SELECT \* FROM issued\_status

WHERE issued\_book\_isbn ='978-0-307-58837-1'

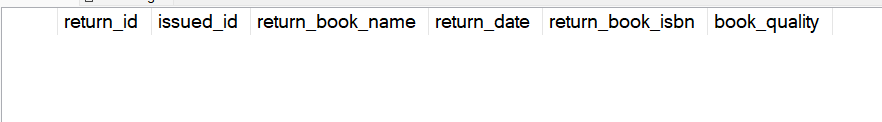


-- Issued\_id IS135

-- Here below as You can see that no return records is there by issue\_id IS135 so book is still with IS135 person

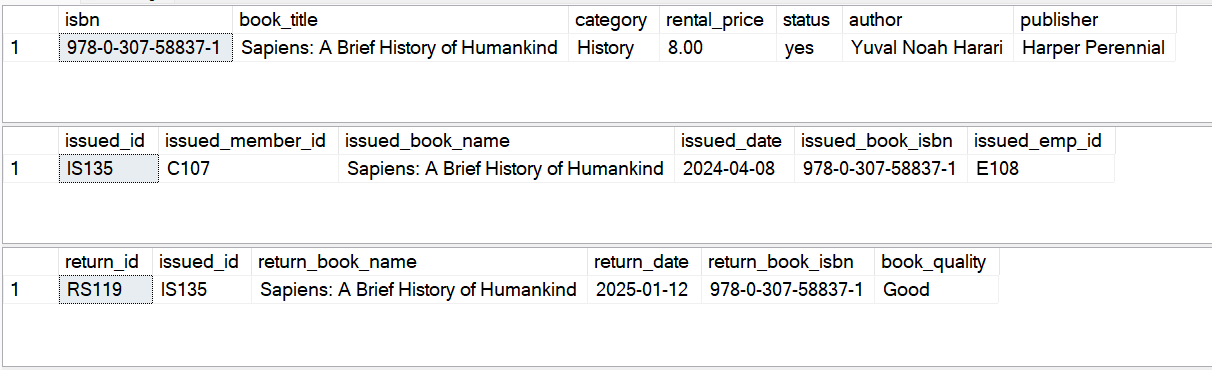
SELECT \* FROM [return\_status ]

WHERE issued\_id = 'IS135'



-- TESTING FUNCTION add\_return\_records

EXEC add\_return\_records 'RS119', 'IS135', 'Good';



-- Testing of add\_return\_records procedure

--UPDATE books

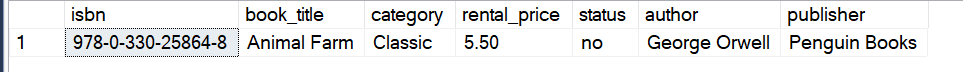
--SET status ='no'

--WHERE isbn = '978-0-330-25864-8'

SELECT \* FROM books

WHERE isbn ='978-0-330-25864-8'

-- Here you can see book is not returned yet

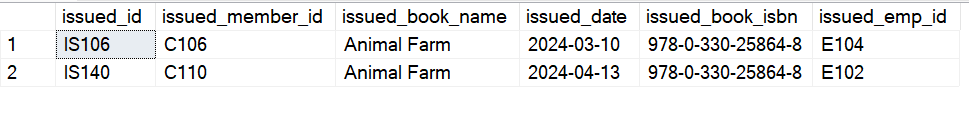


-- By whom this book is issued

SELECT \* FROM issued\_status

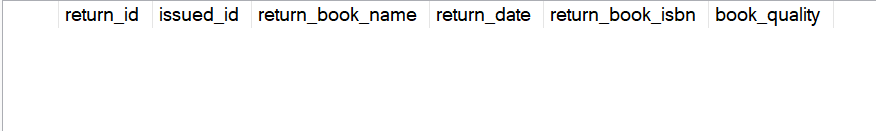
WHERE issued\_book\_isbn ='978-0-330-25864-8'

-- Issued\_id IS140



SELECT \* FROM [return\_status ]

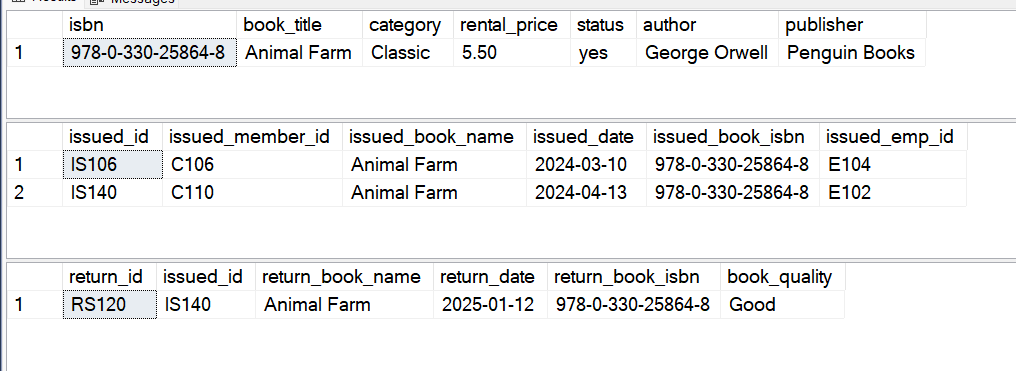
WHERE issued\_id = 'IS140'



--As You can see there is no Return records of IS140

-- TESTING FUNCTION add\_return\_records

EXEC add\_return\_records 'RS120', 'IS140', 'Good';



Task 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.

SELECT

br.branch\_id,

br.manager\_id,

COUNT(isst.issued\_id) Number\_of\_Books\_Issued,

COUNT(rs.return\_id) Number\_of\_Books\_Returned,

SUM(b.rental\_price) Total\_Revenue

INTO Branch\_Report

FROM issued\_status isst

JOIN employees e

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

JOIN branch br

ON br.branch\_id = e.branch\_id

LEFT JOIN [return\_status ] as rs

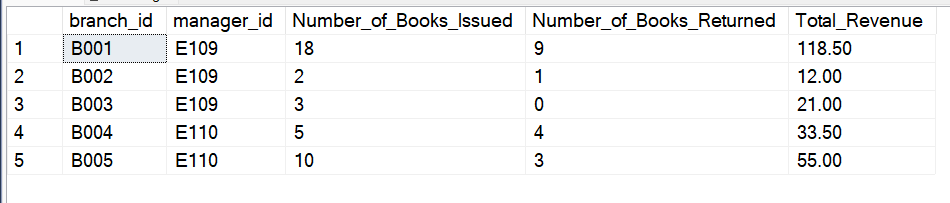
ON rs.issued\_id = isst.issued\_id

JOIN books b

ON isst.issued\_book\_isbn = b.isbn

Group by br.branch\_id, br.manager\_id

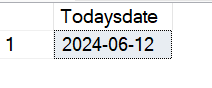
SELECT \* FROM Branch\_Report



Task 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 6 months.

SELECT DATEADD(MONTH,-7,CAST(GETDATE() AS date)) Todaysdate

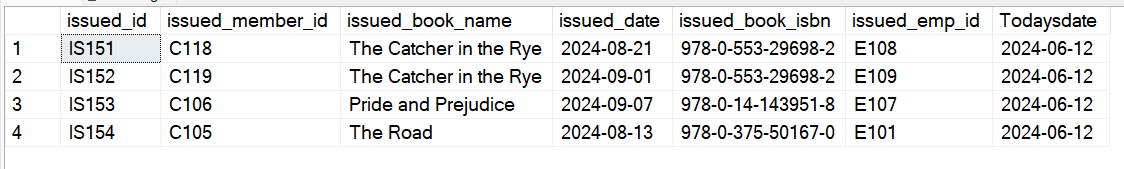


SELECT \*,

DATEADD(MONTH,-7,CAST(GETDATE() AS date)) Todaysdate

FROM issued\_status

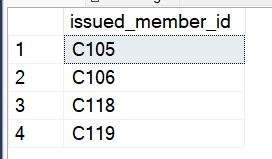
WHERE issued\_date > DATEADD(MONTH,-7,CAST(GETDATE()as date))



SELECT DISTINCT issued\_member\_id

FROM issued\_status

WHERE issued\_date > DATEADD(MONTH,-7,CAST(GETDATE()as date))



SELECT \* INTO Active\_Members FROM members

WHERE member\_id IN

(

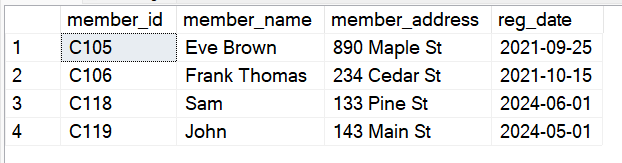
SELECT DISTINCT issued\_member\_id

FROM issued\_status

WHERE issued\_date > DATEADD(MONTH,-7,CAST(GETDATE()as date))

)

SELECT \* FROM Active\_Members



Task 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 TOP 3

e.emp\_name,

br.branch\_id,

br.manager\_id,

branch\_address,

contact\_no,

COUNT(isst.issued\_id) Number\_of\_book\_issued

FROM issued\_status isst

JOIN employees e

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

JOIN branch br

ON e.branch\_id = br.branch\_id

GROUP BY

emp\_name,

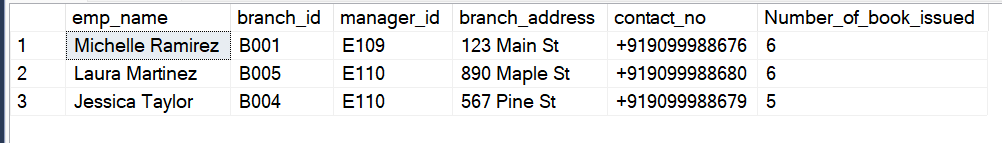
br.branch\_id,

br.manager\_id,

branch\_address,

contact\_no

ORDER BY COUNT(isst.issued\_id) DESC



Task 18: Identify Members Issuing High-Risk Books

Write a query to identify members who have issued books more than twice with the status "damaged" in the books table. Display the member name, book title, and the number of times they've issued damaged books.

SELECT

m.member\_name,

isst.issued\_book\_name,

COUNT(isst.issued\_id) Number\_of\_time\_issued\_book

FROM issued\_status isst

JOIN members m

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

LEFT JOIN [return\_status ] rs

ON rs.issued\_id = isst.issued\_id

WHERE

rs.book\_quality = 'Damaged'

GROUP BY

m.member\_name,

isst.issued\_book\_name

Task 19: 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'.

GO

CREATE PROCEDURE issue\_book

@p\_issued\_id VARCHAR(10),

@p\_issued\_member\_id VARCHAR(30),

@p\_issued\_book\_isbn VARCHAR(50),

@p\_issue\_emp\_id VARCHAR(10)

AS

BEGIN

DECLARE @v\_status VARCHAR(10);

DECLARE @v\_book\_title VARCHAR(100);

-- Check if the book is available

SELECT @v\_status = status,

@v\_book\_title = book\_title

FROM books

WHERE isbn = @p\_issued\_book\_isbn;

-- Handle case where the book is not found

IF @v\_status IS NULL

BEGIN

PRINT 'No book found with the provided ISBN: ' + CAST(@p\_issued\_book\_isbn AS VARCHAR);

RETURN;

END;

-- Check if the book is available

IF @v\_status = 'yes'

BEGIN

-- Insert the issued book record

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

VALUES (@p\_issued\_id, @p\_issued\_member\_id, CAST(GETDATE() AS DATE), @v\_book\_title, @p\_issued\_book\_isbn, @p\_issue\_emp\_id);

PRINT 'Book record added successfully for book ISBN: ' + CAST(@p\_issued\_book\_isbn AS VARCHAR);

-- Update the book status to 'no'

UPDATE books

SET status = 'no'

WHERE isbn = @p\_issued\_book\_isbn;

END

ELSE

BEGIN

PRINT 'Sorry to inform you, the book you have requested is currently unavailable. Book ISBN: ' + CAST(@p\_issued\_book\_isbn AS VARCHAR);

END

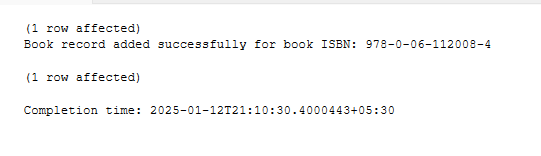
END;

GO

SELECT \* FROM books

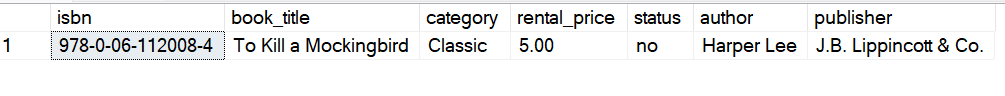
SELECT \* FROM issued\_status

EXEC issue\_book 'IS155','C108','978-0-06-112008-4','E104'

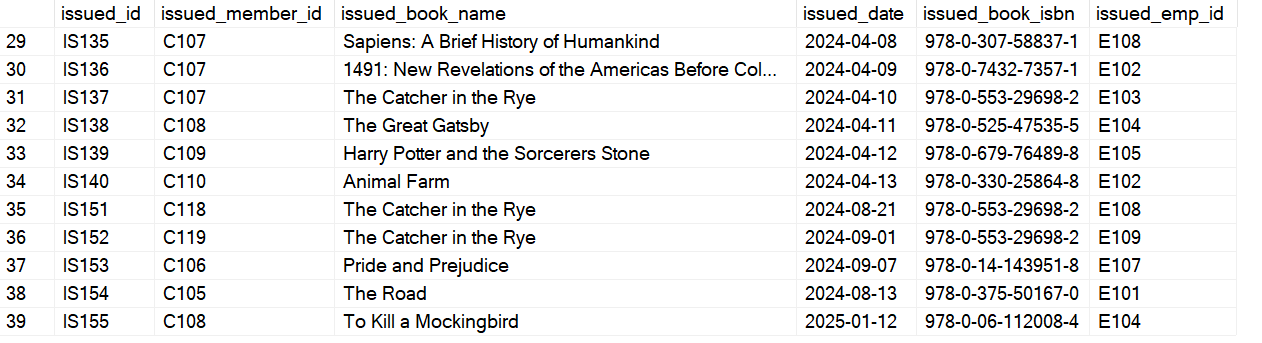


SELECT \* FROM books

WHERE isbn = '978-0-06-112008-4' -- status yes

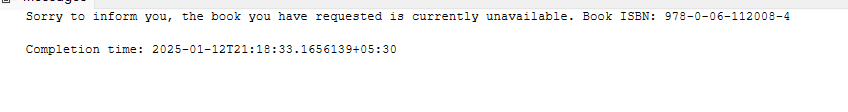


SELECT \* FROM issued\_status



-- Now this book is issued for some now if we another book or suppose this book is issued by some one then

EXEC issue\_book 'IS155','C108','978-0-06-112008-4','E104'



Task 20: Create Table As Select (CTAS)

Objective: Create a CTAS (Create Table As Select) query to identify overdue books and calculate fines.

Description: Write a CTAS query to create a new table that lists each member and the books they have issued but not returned within 30 days. The table should include:

The number of overdue books.

The total fines, with each day's fine calculated at $0.50.

The number of books issued by each member.

The resulting table should show:

Member ID

Number of overdue books

Total fines

\*/

SELECT

m.member\_id,

m.member\_name,

COUNT(isst.issued\_id) AS Number\_of\_Overdue\_books,

COUNT(isst.issued\_id) \* 0.50 AS Total\_Fines

INTO Overdue\_books\_And\_Fines

FROM issued\_status isst

JOIN members m

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

LEFT JOIN [return\_status ] rs

ON rs.issued\_id = isst.issued\_id

JOIN books b

ON b.isbn = isst.issued\_book\_isbn

WHERE

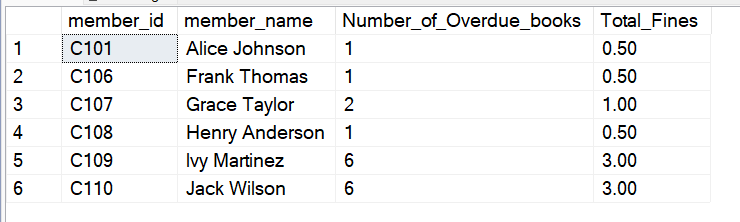
return\_date > DATEADD(DAY,30,issued\_date)

GROUP BY

m.member\_id,

m.member\_name

SELECT \* FROM Overdue\_books\_And\_Fines



**Final ER Diagram of this Project**

