Done By: Noura Al-Mas Al-Maskari

Database Project Part 2

Advanced Queries, Views, and Stored Procedures

1. SQL Script File

--Create Database

CREATE DATABASE LibraryDataBase1;

USE LibraryDataBase1;

--Library Table

CREATE TABLE Library (

LibraryID INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(100) NOT NULL UNIQUE,

Location VARCHAR(150) NOT NULL,

ContactNumber VARCHAR(20) NOT NULL,

EstablishedYear INT

);

--Staff Table

CREATE TABLE Staff (

StaffID INT IDENTITY(1,1) PRIMARY KEY,

FullName VARCHAR(100) NOT NULL,

Position VARCHAR(50) NOT NULL,

ContactNumber VARCHAR(20) NOT NULL,

LibraryID INT NOT NULL,

CONSTRAINT FK_Staff_Library

```sql
FOREIGN KEY (LibraryID)
REFERENCES Library(LibraryID)
ON DELETE CASCADE
ON UPDATE CASCADE
);

--Book Table
CREATE TABLE Book (
BookID INT IDENTITY(1,1) PRIMARY KEY,
ISBN VARCHAR(20) NOT NULL UNIQUE,
Title VARCHAR(150) NOT NULL,
Genre VARCHAR(20) NOT NULL,
Price DECIMAL(10,2) NOT NULL,
ShelfLocation VARCHAR(50) NOT NULL,
IsAvailable BIT NOT NULL DEFAULT (1),
LibraryID INT NOT NULL,
CONSTRAINT CHK_Book_Genre
CHECK (Genre IN ('Fiction','Non-fiction','Reference','Children')),
CONSTRAINT CHK_Book_Price
CHECK (Price > 0),
CONSTRAINT FK_Book_Library
FOREIGN KEY (LibraryID)
REFERENCES Library(LibraryID)
ON DELETE CASCADE
ON UPDATE CASCADE
);

--Member Table
```

```
CREATE TABLE Member (

MemberID INT IDENTITY(1,1) PRIMARY KEY,

FullName VARCHAR(100) NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE,

PhoneNumber VARCHAR(20),

MembershipStartDate DATE NOT NULL

);
```

--Loan Table

```
CREATE TABLE Loan (

LoanID INT IDENTITY(1,1) PRIMARY KEY,

LoanDate DATE NOT NULL,

DueDate DATE NOT NULL,

ReturnDate DATE NULL,

Status VARCHAR(20) NOT NULL DEFAULT ('Issued'),

MemberID INT NOT NULL,

BookID INT NOT NULL,

CONSTRAINT CHK_Loan_Status

CHECK (Status IN ('Issued','Returned','Overdue')),

CONSTRAINT CHK_Loan_ReturnDate

CHECK (ReturnDate IS NULL OR ReturnDate >= LoanDate),

CONSTRAINT FK_Loan_Member

FOREIGN KEY (MemberID)

REFERENCES Member(MemberID)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK_Loan_Book

FOREIGN KEY (BookID)
```

```
REFERENCES Book(BookID)
ON DELETE CASCADE
ON UPDATE CASCADE
);

--Payment Table
CREATE TABLE Payment (
PaymentID INT IDENTITY(1,1) PRIMARY KEY,
PaymentDate DATE NOT NULL,
Amount DECIMAL(10,2) NOT NULL,
Method VARCHAR(50) NOT NULL,
LoanID INT NOT NULL,
CONSTRAINT CHK_Payment_Amount
CHECK (Amount > 0),
CONSTRAINT FK_Payment_Loan
FOREIGN KEY (LoanID)
REFERENCES Loan(LoanID)
ON DELETE CASCADE
ON UPDATE CASCADE
);

--Review Table
CREATE TABLE Review (
ReviewID INT IDENTITY(1,1) PRIMARY KEY,
Rating INT NOT NULL,
Comments NVARCHAR(500) NULL
CONSTRAINT DF_Review_Comments DEFAULT ('No comments'),
ReviewDate DATE NOT NULL,
MemberID INT NOT NULL,

BookID INT NOT NULL,

CONSTRAINT CHK_Review_Rating

CHECK (Rating BETWEEN 1 AND 5),

CONSTRAINT FK_Review_Member

FOREIGN KEY (MemberID)

REFERENCES Member(MemberID)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK_Review_Book

FOREIGN KEY (BookID)

REFERENCES Book(BookID)

ON DELETE CASCADE

ON UPDATE CASCADE

);


--TEST DATA (REQUIRED)...INSERT FOR EACH TABLE

INSERT INTO Library (Name, Location, ContactNumber, EstablishedYear)

VALUES

('Central Library', 'Muscat', '24123456', 2005),

('Nizwa Library', 'Nizwa', '25432111', 2010),

('Sohar Library', 'Sohar', '26889977', 2015);


INSERT INTO Staff (FullName, Position, ContactNumber, LibraryID)

VALUES

('Ahmed Al-Harthy', 'Manager', '91234567', 1),

('Fatma Al-Zadjali', 'Assistant', '92345678', 2),

('Salim Al-Rawahi', 'Librarian', '93456789', 3);
```

```
INSERT INTO Member (FullName, Email, PhoneNumber, MembershipStartDate)
VALUES
('Ali Said', 'ali@mail.com', '90000001', '2024-01-01'),
('Muna Rashid', 'muna@mail.com', '90000002', '2024-01-05'),
('Khalid Ahmed', 'khalid@mail.com', '90000003', '2024-02-01'),
('Aisha Noor', 'aisha@mail.com', '90000004', '2024-02-10'),
('Salma Hassan', 'salma@mail.com', '90000005', '2024-03-01'),
('Yousef Omar', 'yousef@mail.com', '90000006', '2024-03-15'),
('Noor Said', 'noor@mail.com', '90000007', '2024-04-01'),
('Hamed Ali', 'hamed@mail.com', '90000008', '2024-04-05'),
('Sara Nasser', 'sara@mail.com', '90000009', '2024-04-10'),
('Omar Khalfan', 'omar@mail.com', '90000010', '2024-05-01');


INSERT INTO Book (ISBN, Title, Genre, Price, ShelfLocation, LibraryID)
VALUES
('ISBN001','SQL Basics','Reference',15,'A1',1),
('ISBN002','Advanced SQL','Reference',20,'A2',1),
('ISBN003','Database Design','Reference',18,'A3',1),
('ISBN004','Harry Potter','Fiction',10,'B1',1),
('ISBN005','The Hobbit','Fiction',12,'B2',1),

('ISBN006','Data Science','Non-fiction',22,'C1',2),
('ISBN007','AI Basics','Non-fiction',25,'C2',2),
('ISBN008','Python Guide','Reference',19,'C3',2),
('ISBN009','Fairy Tales','Children',8,'D1',2),
```

```sql
('ISBN010','Kids Math','Children',6,'D2',2),


('ISBN011','History of Oman','Non-fiction',14,'E1',3),
('ISBN012','World Atlas','Reference',30,'E2',3),
('ISBN013','Science Fun','Children',9,'E3',3),
('ISBN014','Mystery Island','Fiction',11,'F1',3),
('ISBN015','Lost City','Fiction',13,'F2',3),
('ISBN016','Novel A','Fiction',10,'F3',3),
('ISBN017','Novel B','Fiction',10,'F4',3),
('ISBN018','Novel C','Fiction',10,'F5',3),
('ISBN019','Novel D','Fiction',10,'F6',3),
('ISBN020','Novel E','Fiction',10,'F7',3);

INSERT INTO Loan (LoanDate, DueDate, Status, MemberID, BookID)
VALUES
('2024-05-01','2024-05-10','Returned',1,1),
('2024-05-03','2024-05-12','Returned',2,2),
('2024-05-05','2024-05-15','Returned',3,3),
('2024-05-07','2024-05-17','Overdue',4,4),
('2024-05-09','2024-05-19','Issued',5,5),
('2024-05-10','2024-05-20','Returned',6,6),
('2024-05-11','2024-05-21','Returned',7,7),
('2024-05-12','2024-05-22','Issued',8,8),
('2024-05-13','2024-05-23','Returned',9,9),
('2024-05-14','2024-05-24','Returned',10,10),
('2024-06-01','2024-06-10','Issued',1,11),
('2024-06-02','2024-06-11','Issued',2,12),
('2024-06-03','2024-06-12','Returned',3,13),
('2024-06-04','2024-06-13','Returned',4,14),
('2024-06-05','2024-06-14','Issued',5,15);



INSERT INTO Payment (PaymentDate, Amount, Method, LoanID)
VALUES
(GETDATE(), 6, 'Cash', 4),
(GETDATE(), 4, 'Card', 1),
(GETDATE(), 8, 'Cash', 7);


--Section 1: Complex Queries with Joins
--1.Library Book Inventory Report

SELECT l.Name,
COUNT(b.BookID) TotalBooks,
SUM(CASE WHEN b.IsAvailable=1 THEN 1 ELSE 0 END) AvailableBooks,
SUM(CASE WHEN b.IsAvailable=0 THEN 1 ELSE 0 END) BooksOnLoan
FROM Library l
LEFT JOIN Book b ON l.LibraryID=b.LibraryID
GROUP BY l.Name;


--2. Active Borrowers Analysis

SELECT m.FullName, m.Email, b.Title, ln.LoanDate, ln.DueDate, ln.Status
```

```
FROM Loan ln

JOIN Member m ON ln.MemberID=m.MemberID

JOIN Book b ON ln.BookID=b.BookID

WHERE ln.Status IN ('Issued','Overdue');



--3. Overdue Loans with Member Details


SELECT m.FullName, m.PhoneNumber, b.Title, l.Name Library,

DATEDIFF(DAY, ln.DueDate, GETDATE()) DaysOverdue,

ISNULL(SUM(p.Amount),0) FinesPaid

FROM Loan ln

JOIN Member m ON ln.MemberID=m.MemberID

JOIN Book b ON ln.BookID=b.BookID

JOIN Library l ON b.LibraryID=l.LibraryID

LEFT JOIN Payment p ON ln.LoanID=p.LoanID

WHERE ln.Status='Overdue'

GROUP BY m.FullName,m.PhoneNumber,b.Title,l.Name,ln.DueDate;



--4. Staff Performance Overview

SELECT l.Name, s.FullName, s.Position, COUNT(b.BookID) BooksManaged

FROM Library l

JOIN Staff s ON l.LibraryID=s.LibraryID

LEFT JOIN Book b ON l.LibraryID=b.LibraryID

GROUP BY l.Name,s.FullName,s.Position;
```

```sql
--5. Book Popularity Report
SELECT b.Title,b.ISBN,b.Genre,
COUNT(ln.LoanID) TimesLoaned,
AVG(r.Rating) AvgRating
FROM Book b
JOIN Loan ln ON b.BookID=ln.BookID
LEFT JOIN Review r ON b.BookID=r.BookID
GROUP BY b.Title,b.ISBN,b.Genre
HAVING COUNT(ln.LoanID)>=3;


--6. Member Reading History
SELECT m.FullName,b.Title,ln.LoanDate,ln.ReturnDate,r.Rating,r.Comments
FROM Member m
JOIN Loan ln ON m.MemberID=ln.MemberID
JOIN Book b ON ln.BookID=b.BookID
LEFT JOIN Review r ON r.MemberID=m.MemberID AND r.BookID=b.BookID;


--7. Revenue Analysis by Genre
SELECT b.Genre,
COUNT(ln.LoanID) TotalLoans,
SUM(p.Amount) TotalRevenue,
AVG(p.Amount) AvgFine
FROM Book b
JOIN Loan ln ON b.BookID=ln.BookID
JOIN Payment p ON ln.LoanID=p.LoanID

GROUP BY b.Genre;
```

--Section 2: Aggregate Functions and Grouping

```sql
--8. Monthly Loan Statistics

SELECT

DATENAME(MONTH, LoanDate) AS MonthName,

COUNT(*) AS TotalLoans,

SUM(CASE WHEN Status = 'Returned' THEN 1 ELSE 0 END) AS Returned,

SUM(CASE WHEN Status IN ('Issued','Overdue') THEN 1 ELSE 0 END) AS ActiveLoans

FROM Loan

WHERE YEAR(LoanDate) = YEAR(GETDATE())

GROUP BY DATENAME(MONTH, LoanDate), MONTH(LoanDate)

ORDER BY MONTH(LoanDate);


--9. Member Engagement Metrics

SELECT

m.FullName,

COUNT(ln.LoanID) AS TotalBorrowed,

SUM(CASE WHEN ln.Status IN ('Issued','Overdue') THEN 1 ELSE 0 END) AS
CurrentlyBorrowed,

ISNULL(SUM(p.Amount),0) AS TotalFines,

AVG(r.Rating) AS AvgRating

FROM Member m

JOIN Loan ln ON m.MemberID = ln.MemberID

LEFT JOIN Payment p ON ln.LoanID = p.LoanID

LEFT JOIN Review r ON m.MemberID = r.MemberID
```

```
GROUP BY m.FullName;

--10. Library Performance Comparison
SELECT
l.Name,
COUNT(b.BookID) AS TotalBooks,
COUNT(ln.MemberID) AS ActiveMembers,
ISNULL(SUM(p.Amount),0) AS TotalRevenue,
COUNT(b.BookID) * 1.0 / NULLIF(COUNT(ln.MemberID),0) AS AvgBooksPerMember
FROM Library l
LEFT JOIN Book b ON l.LibraryID = b.LibraryID
LEFT JOIN Loan ln ON b.BookID = ln.BookID
LEFT JOIN Payment p ON ln.LoanID = p.LoanID
GROUP BY l.Name;


--11. High-Value Books Analysis
SELECT
b.Title,
b.Genre,
b.Price,
AVG(b2.Price) OVER (PARTITION BY b.Genre) AS GenreAvgPrice,
b.Price - AVG(b2.Price) OVER (PARTITION BY b.Genre) AS PriceDifference
FROM Book b
JOIN Book b2 ON b.Genre = b2.Genre
WHERE b.Price > (
SELECT AVG(Price)
FROM Book
WHERE Genre = b.Genre
);

--12. Payment Pattern Analysis

SELECT
Method,
COUNT(*) AS Transactions,
SUM(Amount) AS TotalCollected,
AVG(Amount) AS AvgPayment,
SUM(Amount) * 100.0 / (SELECT SUM(Amount) FROM Payment) AS PercentageOfRevenue
FROM Payment
GROUP BY Method;
```

--Section 3: Views Creation

```
--13. vw_CurrentLoans
CREATE VIEW vw_CurrentLoans AS
SELECT
m.FullName,
b.Title,
ln.LoanDate,
ln.DueDate,
ln.Status,
DATEDIFF(DAY, GETDATE(), ln.DueDate) AS DaysToOrOverdue
FROM Loan ln
```

```sql
JOIN Member m ON ln.MemberID = m.MemberID

JOIN Book b ON ln.BookID = b.BookID

WHERE ln.Status IN ('Issued','Overdue');


--14. vw_LibraryStatistics


--(Q14- VW_LIBRARYSTATISTICS SHOULD SHOW LIBRARY NAME,

--TOTAL BOOKS OWNED BY THE LIBRARY,

--NUMBER OF AVAILABLE BOOKS,

--TOTAL ACTIVE MEMBERS(MEMBERS WHO HAVE AT LEAST ONE LOAN FROM THIS
LIBRARY'S BOOKS),

--ACTIVE LOANS (LOANS OF BOOKS BELONGING TO THIS LIBRARY ),

--TOTAL STAFF WORKING AT THE LIBRARY,

--TOTAL REVENUE FROM FINES (FROM LOANS OF THIS LIBRARY'S BOOKS).)


CREATE VIEW vw_LibraryStatistics AS


SELECT

l.Name AS LibraryName,

COUNT(b.BookID) AS TotalBooksOwned,

SUM(CASE WHEN b.IsAvailable = 1 THEN 1 ELSE 0 END) AS AvailableBooks,

COUNT(DISTINCT ln.MemberID) AS ActiveMembers,

COUNT(DISTINCT ln.LoanID) AS ActiveLoans,

COUNT(DISTINCT s.StaffID) AS TotalStaff,

ISNULL(SUM(p.Amount), 0) AS TotalRevenue

FROM Library l

LEFT JOIN Book b ON l.LibraryID = b.LibraryID

LEFT JOIN Loan ln ON b.BookID = ln.BookID AND ln.Status IN ('Issued', 'Overdue')
LEFT JOIN Staff s ON l.LibraryID = s.LibraryID

LEFT JOIN Payment p ON ln.LoanID = p.LoanID

GROUP BY l.Name;


--15. vw_BookDetailsWithReviews


CREATE VIEW vw_BookDetailsWithReviews AS

SELECT

b.Title,

b.Genre,

b.IsAvailable,

AVG(r.Rating) AS AvgRating,

COUNT(r.ReviewID) AS TotalReviews,

MAX(r.ReviewDate) AS LatestReview

FROM Book b

LEFT JOIN Review r ON b.BookID = r.BookID

GROUP BY b.Title, b.Genre, b.IsAvailable;
```

--Section 4: Stored Procedures

```
--16. sp_IssueBook

CREATE PROCEDURE sp_IssueBook

@MemberID INT,

@BookID INT,

@DueDate DATE

AS

BEGIN
```

```
SET NOCOUNT ON;
BEGIN TRANSACTION;
BEGIN TRY
-- Check book availability
IF NOT EXISTS (
SELECT 1 FROM Book
WHERE BookID = @BookID AND IsAvailable = 1
)
BEGIN
THROW 50001, 'Book is not available.', 1;
END
-- Check overdue loans
IF EXISTS (
SELECT 1 FROM Loan
WHERE MemberID = @MemberID AND Status = 'Overdue'
)
BEGIN
THROW 50002, 'Member has overdue loans.', 1;
END
-- Create loan
INSERT INTO Loan (LoanDate, DueDate, Status, MemberID, BookID)
VALUES (GETDATE(), @DueDate, 'Issued', @MemberID, @BookID);
-- Update book availability
UPDATE Book
SET IsAvailable = 0
WHERE BookID = @BookID;
COMMIT TRANSACTION;
SELECT 'Book issued successfully.' AS Message;
```

```
END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

SELECT ERROR_MESSAGE() AS ErrorMessage;

END CATCH

END;


--Test Cases

--Successful

EXEC sp_IssueBook 1, 3, '2025-02-01';


-- Error

EXEC sp_IssueBook 1, 3, '2025-02-01';




--17. sp_ReturnBook

CREATE PROCEDURE sp_ReturnBook

@LoanID INT,

@ReturnDate DATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @DueDate DATE, @BookID INT, @DaysLate INT, @Fine DECIMAL(10,2);

BEGIN TRANSACTION;

BEGIN TRY

SELECT

@DueDate = DueDate,
@BookID = BookID
FROM Loan
WHERE LoanID = @LoanID AND Status <> 'Returned';
IF @DueDate IS NULL
THROW 50003, 'Invalid or already returned loan.', 1;
-- Calculate fine
SET @DaysLate = DATEDIFF(DAY, @DueDate, @ReturnDate);
SET @Fine = CASE WHEN @DaysLate > 0 THEN @DaysLate * 2 ELSE 0 END;
-- Update loan
UPDATE Loan
SET ReturnDate = @ReturnDate,
Status = 'Returned'
WHERE LoanID = @LoanID;
-- Update book
UPDATE Book
SET IsAvailable = 1
WHERE BookID = @BookID;
-- Create payment if fine exists
IF @Fine > 0
BEGIN
INSERT INTO Payment (PaymentDate, Amount, Method, LoanID)
VALUES (GETDATE(), @Fine, 'Pending', @LoanID);
END
COMMIT TRANSACTION;
SELECT @Fine AS TotalFine;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION;
```

```
SELECT ERROR_MESSAGE() AS ErrorMessage;
END CATCH
END;

--Test Cases
--Successful
EXEC sp_ReturnBook 5, '2025-01-20';

-- Error
EXEC sp_ReturnBook 5, '2025-01-25';



--18. sp_GetMemberReport
CREATE  PROCEDURE sp_GetMemberReport
@MemberID INT
AS
BEGIN
-- Member info
SELECT * FROM Member WHERE MemberID = @MemberID;
-- Current loans
SELECT * FROM Loan
WHERE MemberID = @MemberID AND Status IN ('Issued','Overdue');
-- Loan history
SELECT * FROM Loan
WHERE MemberID = @MemberID;
-- Fines summary
SELECT
SUM(CASE WHEN Method <> 'Pending' THEN Amount ELSE 0 END) AS TotalPaid,
SUM(CASE WHEN Method = 'Pending' THEN Amount ELSE 0 END) AS PendingFines

FROM Payment p

JOIN Loan l ON p.LoanID = l.LoanID

WHERE l.MemberID = @MemberID;

-- Reviews

SELECT * FROM Review WHERE MemberID = @MemberID;

END;
```

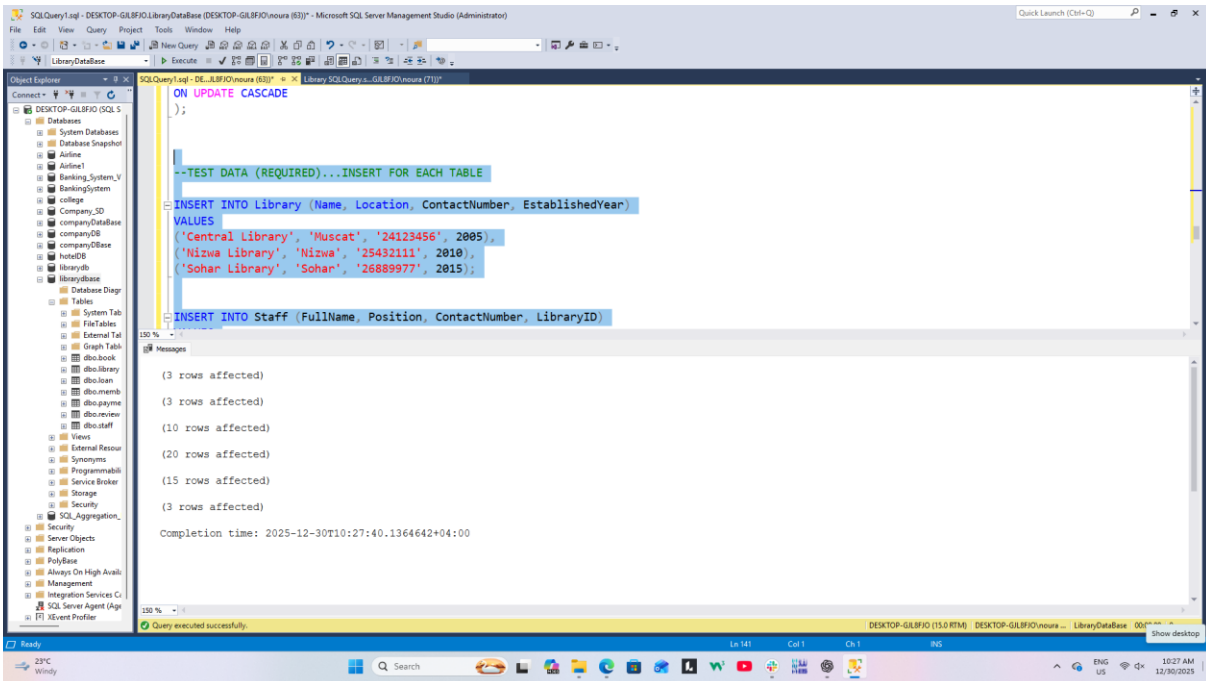
--Test Cases

```
--Successful

EXEC sp_GetMemberReport 1;


-- Error

EXEC sp_GetMemberReport 9;


--19. sp_MonthlyLibraryReport

CREATE  PROCEDURE sp_MonthlyLibraryReport

@LibraryID INT,

@Month INT,

@Year INT

AS

BEGIN

-- Total loans issued

SELECT COUNT(*) AS TotalLoansIssued
```

```
FROM Loan l
JOIN Book b ON l.BookID = b.BookID
WHERE b.LibraryID = @LibraryID
AND MONTH(l.LoanDate) = @Month
AND YEAR(l.LoanDate) = @Year;
-- Total returns
SELECT COUNT(*) AS TotalBooksReturned
FROM Loan l
JOIN Book b ON l.BookID = b.BookID
WHERE b.LibraryID = @LibraryID
AND MONTH(l.ReturnDate) = @Month
AND YEAR(l.ReturnDate) = @Year;
-- Total revenue
SELECT ISNULL(SUM(p.Amount),0) AS TotalRevenue
FROM Payment p
JOIN Loan l ON p.LoanID = l.LoanID
JOIN Book b ON l.BookID = b.BookID
WHERE b.LibraryID = @LibraryID
AND MONTH(p.PaymentDate) = @Month
AND YEAR(p.PaymentDate) = @Year;
-- Most borrowed genre
SELECT TOP 1 b.Genre, COUNT(*) AS BorrowCount
FROM Loan l
JOIN Book b ON l.BookID = b.BookID
WHERE b.LibraryID = @LibraryID
GROUP BY b.Genre
ORDER BY BorrowCount DESC;
-- Top 3 active members
```

```
SELECT TOP 3 m.FullName, COUNT(*) AS LoanCount

FROM Loan l

JOIN Member m ON l.MemberID = m.MemberID

JOIN Book b ON l.BookID = b.BookID

WHERE b.LibraryID = @LibraryID

GROUP BY m.FullName

ORDER BY LoanCount DESC;

END;




--Test Cases

--Successful

EXEC sp_MonthlyLibraryReport 1, 1, 2025;


-- Error(Month with no activity)

EXEC sp_MonthlyLibraryReport 1, 12, 2030;
```

*Test data inserts demonstrating that your procedures work correctly



```
ON UPDATE CASCADE
);


--TEST DATA (REQUIRED)...INSERT FOR EACH TABLE

INSERT INTO Library (Name, Location, ContactNumber, EstablishedYear)
VALUES
('Central Library', 'Muscat', '24123456', 2005),
('Nizwa Library', 'Nizwa', '25432111', 2010),
('Sohar Library', 'Sohar', '26889977', 2015);


INSERT INTO Staff (FullName, Position, ContactNumber, LibraryID)
```

```
(3 rows affected)

(3 rows affected)

(10 rows affected)

(20 rows affected)

(15 rows affected)

(3 rows affected)

Completion time: 2025-12-30T10:27:40.1364642+04:00
```

Query executed successfully.

2. Documentation Document

Including:

• Brief explanation of your approach for complex queries (queries 5, 6, 7)

Q5: Book Popularity Report

Identify books that are frequently borrowed (at least 3 times) and evaluate reader feedback.

How I did it:

- Joined **Book**, **Loan**, and **Review** tables.
- Used `COUNT(LoanID)` to calculate how many times each book was loaned.
- Used `AVG(Rating)` to compute the average review rating per book.
- Applied `HAVING COUNT(LoanID) >= 3` to filter popular books only.
- Used `LEFT JOIN` for reviews to ensure books without reviews are still included.

This approach efficiently combines transactional data (loans) with feedback data (reviews) to provide a meaningful popularity metric without excluding incomplete data.

Q6: Member Reading History
Show a complete borrowing history for each member, including reviews.

How I did it:

- Joined **Member → Loan → Book** to retrieve borrowing records.
- Used a `LEFT JOIN` with **Review** to include review information only when it exists.
- Displayed both returned and currently borrowed books.
- Sorted by member name and loan date for readability.

Using `LEFT JOIN` ensures no borrowing record is lost, even if the member did not leave a review, giving a full historical view.

Q7: Revenue Analysis by Genre
Analyze fine revenue collected per book genre.

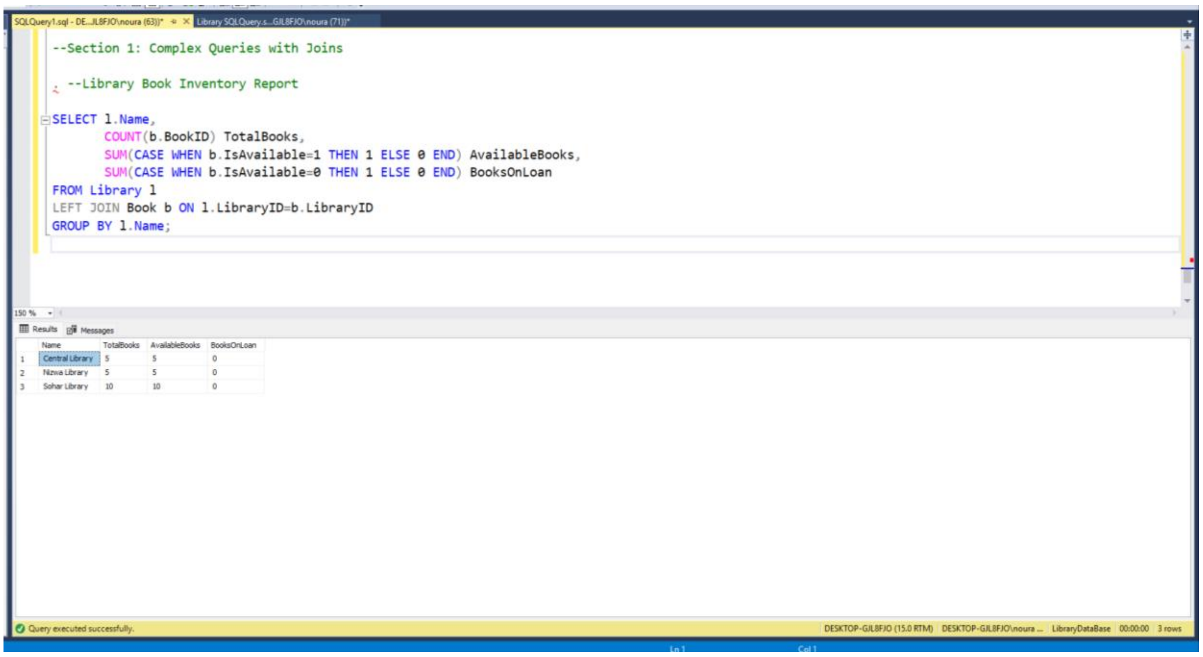
How i did it:

- Joined **Book**, **Loan**, and **Payment** tables.
- Grouped data by book genre.
- Used aggregate functions:
  - `COUNT(DISTINCT LoanID)` for total loans
  - `SUM(Amount)` for total revenue
  - `AVG(Amount)` for average fine per loan

This approach links financial data to book categories, allowing management to identify which genres generate higher fines and borrowing activity.

- **Screenshots of query results for at least 5 different queries**

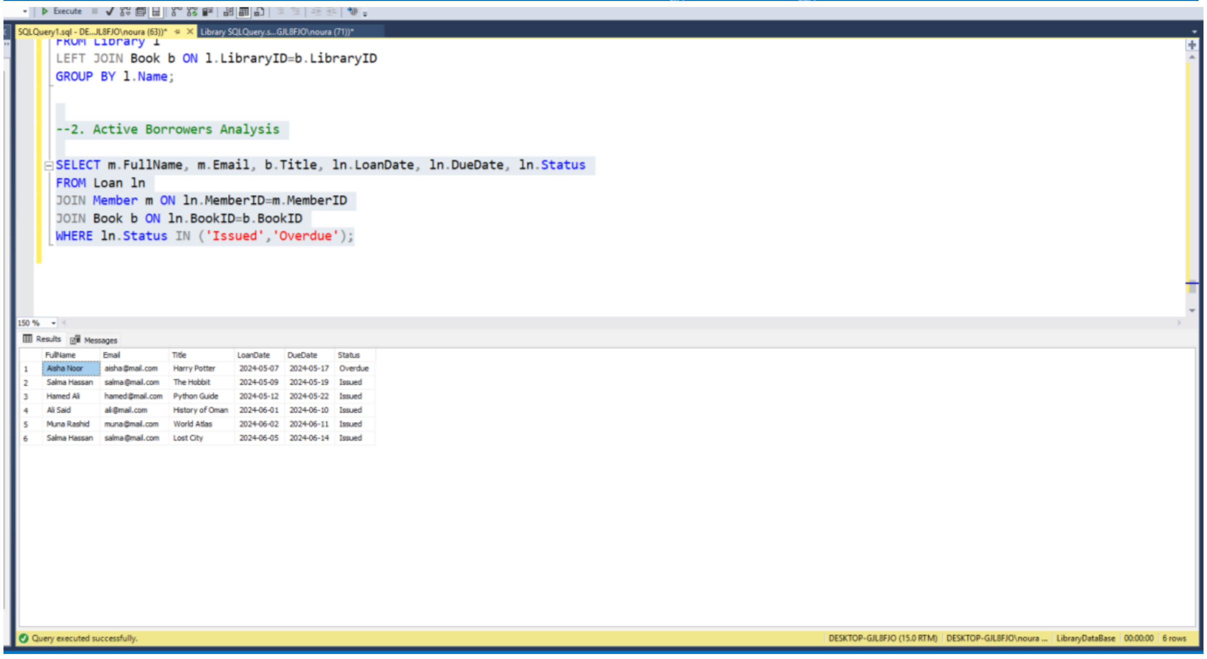
Section 1: Complex Queries with Joins



```sql
--Section 1: Complex Queries with Joins

 --Library Book Inventory Report

SELECT l.Name,
       COUNT(b.BookID) TotalBooks,
       SUM(CASE WHEN b.IsAvailable=1 THEN 1 ELSE 0 END) AvailableBooks,
       SUM(CASE WHEN b.IsAvailable=0 THEN 1 ELSE 0 END) BooksOnLoan
FROM Library l
LEFT JOIN Book b ON l.LibraryID=b.LibraryID
GROUP BY l.Name;
```

	Name	TotalBooks	AvailableBooks	BooksOnLoan
1	Central Library	5	5	0
2	Nizwa Library	5	5	0
3	Sohar Library	10	10	0

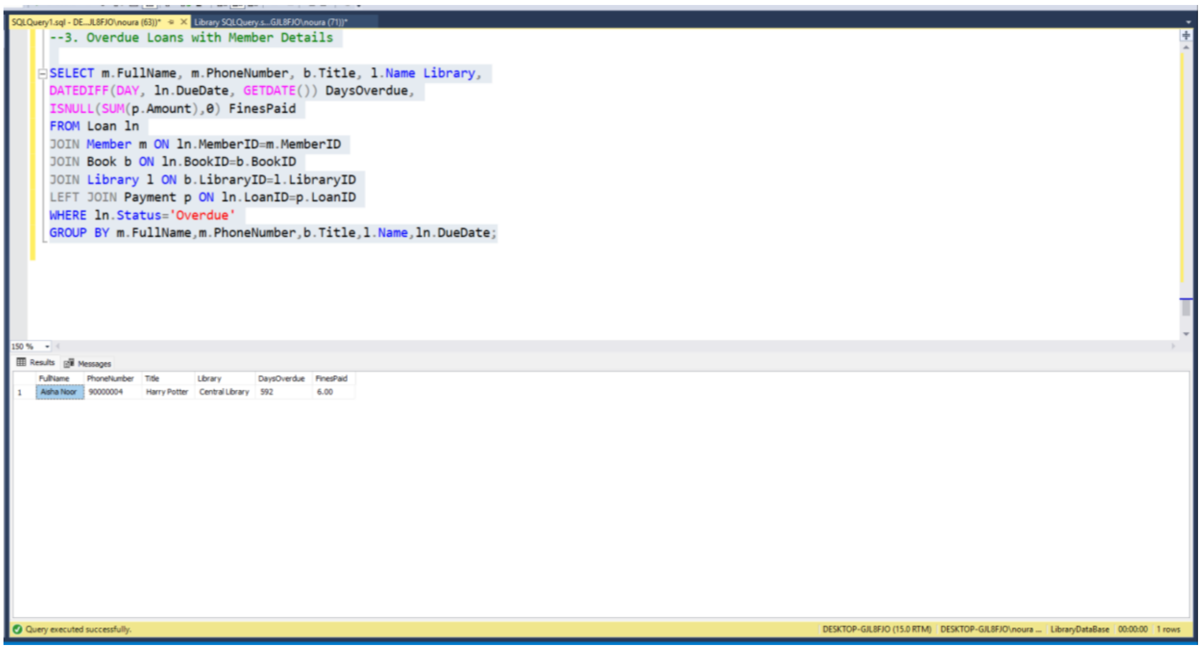
Query executed successfully. DESKTOP-GJL8FJO (15.0 RTM) DESKTOP-GJL8FJO\noura ... LibraryDataBase 00:00:00 3 rows



```sql
FROM Library l
LEFT JOIN Book b ON l.LibraryID=b.LibraryID
GROUP BY l.Name;


--2. Active Borrowers Analysis

SELECT m.FullName, m.Email, b.Title, ln.LoanDate, ln.DueDate, ln.Status
FROM Loan ln
JOIN Member m ON ln.MemberID=m.MemberID
JOIN Book b ON ln.BookID=b.BookID
WHERE ln.Status IN ('Issued','Overdue');
```

	FullName	Email	Title	LoanDate	DueDate	Status
1	Aisha Noor	aisha@mail.com	Harry Potter	2024-05-07	2024-05-17	Overdue
2	Salma Hassan	salma@mail.com	The Hobbit	2024-05-09	2024-05-19	Issued
3	Hamed Ali	hamed@mail.com	Python Guide	2024-05-12	2024-05-22	Issued
4	Ali Said	ali@mail.com	History of Oman	2024-06-01	2024-06-10	Issued
5	Muna Rashid	muna@mail.com	World Atlas	2024-06-02	2024-06-11	Issued
6	Salma Hassan	salma@mail.com	Lost City	2024-06-05	2024-06-14	Issued

Query executed successfully. DESKTOP-GJL8FJO (15.0 RTM) DESKTOP-GJL8FJO\noura ... LibraryDataBase 00:00:00 6 rows

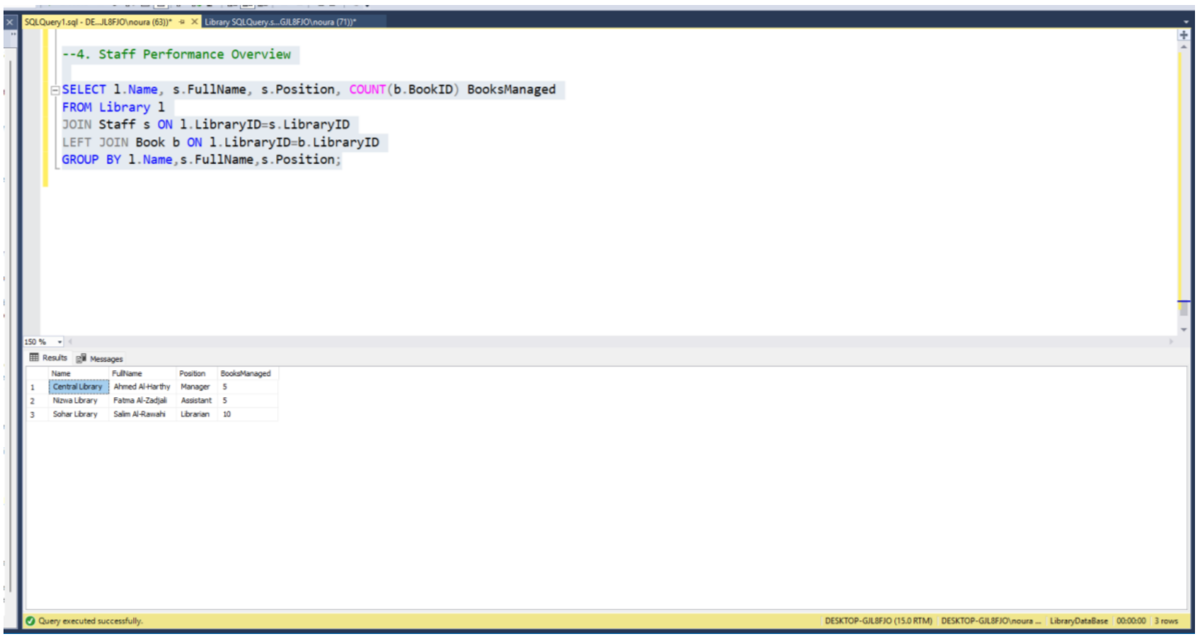


SQLQuery1.sql - DE...JL8FJO\noura (63))*
Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--3. Overdue Loans with Member Details

SELECT m.FullName, m.PhoneNumber, b.Title, l.Name Library,
DATEDIFF(DAY, ln.DueDate, GETDATE()) DaysOverdue,
ISNULL(SUM(p.Amount),0) FinesPaid
FROM Loan ln
JOIN Member m ON ln.MemberID=m.MemberID
JOIN Book b ON ln.BookID=b.BookID
JOIN Library l ON b.LibraryID=l.LibraryID
LEFT JOIN Payment p ON ln.LoanID=p.LoanID
WHERE ln.Status='Overdue'
GROUP BY m.FullName,m.PhoneNumber,b.Title,l.Name,ln.DueDate;
```

	FullName	PhoneNumber	Title	Library	DaysOverdue	FinesPaid
1	Aisha Noor	90000004	Harry Potter	Central Library	592	6.00

```
Query executed successfully.   DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 1 rows
```

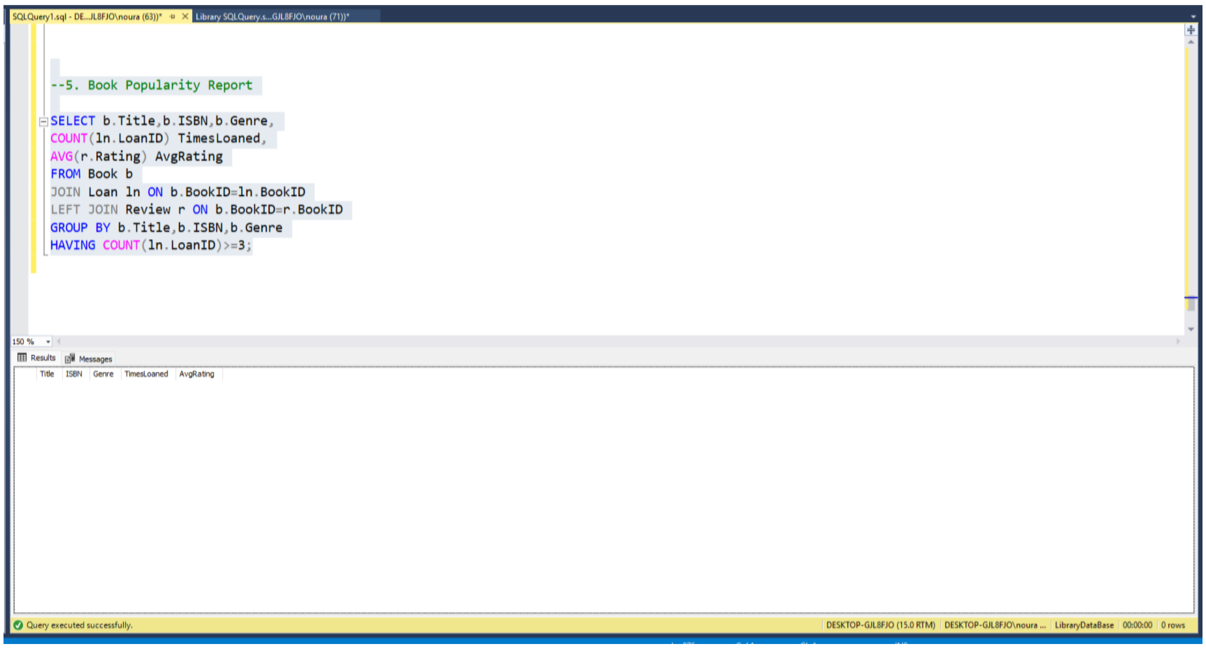


SQLQuery1.sql - DE...JL8FJO\noura (63))*
Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--4. Staff Performance Overview

SELECT l.Name, s.FullName, s.Position, COUNT(b.BookID) BooksManaged
FROM Library l
JOIN Staff s ON l.LibraryID=s.LibraryID
LEFT JOIN Book b ON l.LibraryID=b.LibraryID
GROUP BY l.Name,s.FullName,s.Position;
```

	Name	FullName	Position	BooksManaged
1	Central Library	Ahmed Al-Harthy	Manager	5
2	Nizwa Library	Fatma Al-Zadjali	Assistant	5
3	Sohar Library	Salim Al-Rawahi	Librarian	10

```
Query executed successfully.   DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 3 rows
```



SQLQuery1.sql - DE...JL8FJO\noura (63))* Library SQLQuery.s...GJL8FJO\noura (71))*

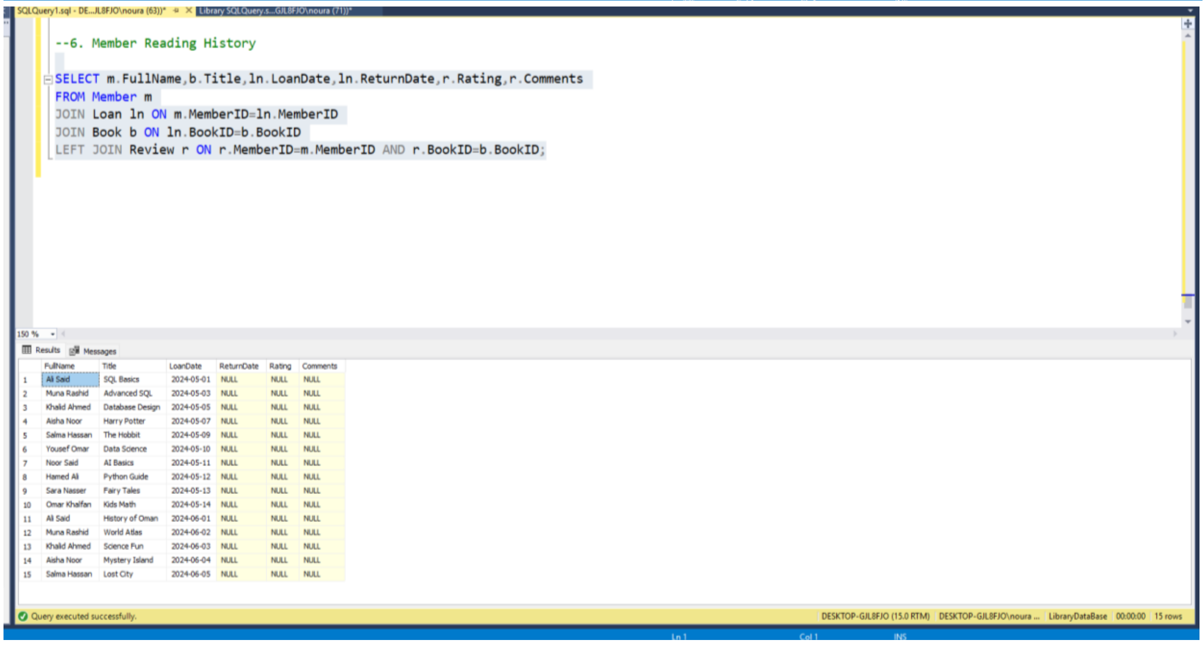
```sql
--5. Book Popularity Report

SELECT b.Title,b.ISBN,b.Genre,
COUNT(ln.LoanID) TimesLoaned,
AVG(r.Rating) AvgRating
FROM Book b
JOIN Loan ln ON b.BookID=ln.BookID
LEFT JOIN Review r ON b.BookID=r.BookID
GROUP BY b.Title,b.ISBN,b.Genre
HAVING COUNT(ln.LoanID)>=3;
```

150 %

Results Messages

	Title	ISBN	Genre	TimesLoaned	AvgRating

Query executed successfully. DESKTOP-GJL8FJO (15.0 RTM) DESKTOP-GJL8FJO\noura ... LibraryDataBase 00:00:00 0 rows



SQLQuery1.sql - DE...JL8FJO\noura (63))* Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--6. Member Reading History

SELECT m.FullName,b.Title,ln.LoanDate,ln.ReturnDate,r.Rating,r.Comments
FROM Member m
JOIN Loan ln ON m.MemberID=ln.MemberID
JOIN Book b ON ln.BookID=b.BookID
LEFT JOIN Review r ON r.MemberID=m.MemberID AND r.BookID=b.BookID;
```

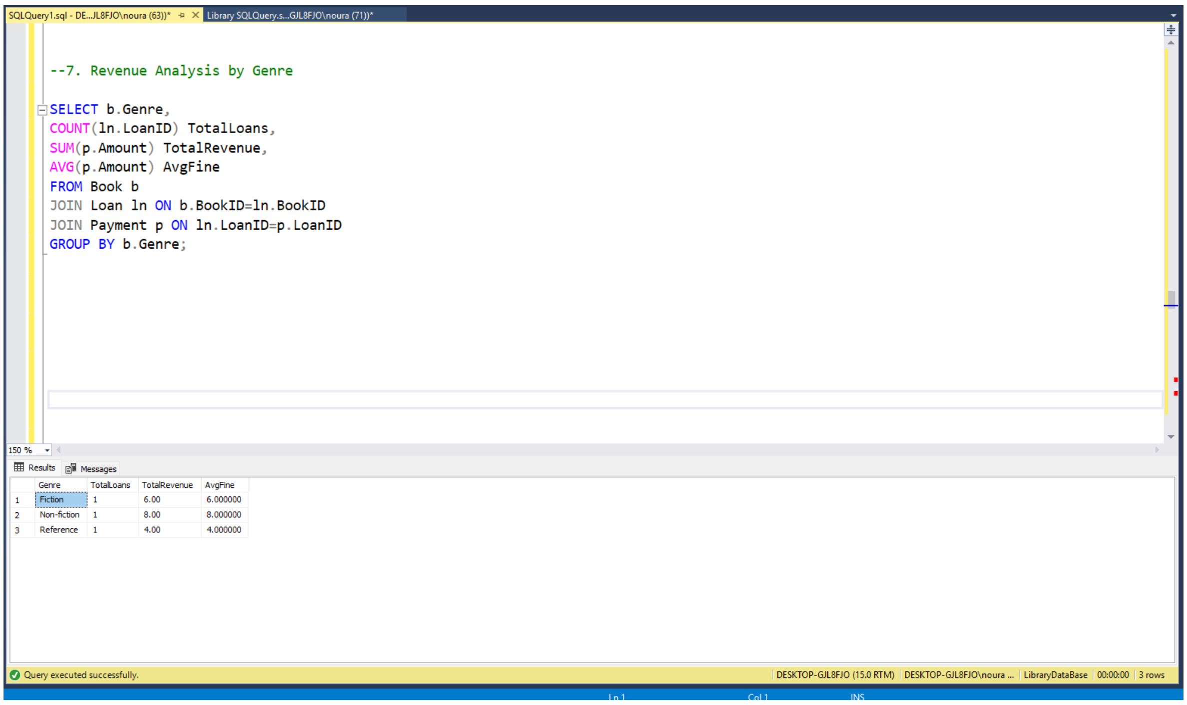
150 %

Results Messages

	FullName	Title	LoanDate	ReturnDate	Rating	Comments
1	Ali Said	SQL Basics	2024-05-01	NULL	NULL	NULL
2	Muna Rashid	Advanced SQL	2024-05-03	NULL	NULL	NULL
3	Khalid Ahmed	Database Design	2024-05-05	NULL	NULL	NULL
4	Aisha Noor	Harry Potter	2024-05-07	NULL	NULL	NULL
5	Salma Hassan	The Hobbit	2024-05-09	NULL	NULL	NULL
6	Yousef Omar	Data Science	2024-05-10	NULL	NULL	NULL
7	Noor Said	AI Basics	2024-05-11	NULL	NULL	NULL
8	Hamed Ali	Python Guide	2024-05-12	NULL	NULL	NULL
9	Sara Nasser	Fairy Tales	2024-05-13	NULL	NULL	NULL
10	Omar Khalfan	Kids Math	2024-05-14	NULL	NULL	NULL
11	Ali Said	History of Oman	2024-06-01	NULL	NULL	NULL
12	Muna Rashid	World Atlas	2024-06-02	NULL	NULL	NULL
13	Khalid Ahmed	Science Fun	2024-06-03	NULL	NULL	NULL
14	Aisha Noor	Mystery Island	2024-06-04	NULL	NULL	NULL
15	Salma Hassan	Lost City	2024-06-05	NULL	NULL	NULL

Query executed successfully. DESKTOP-GJL8FJO (15.0 RTM) DESKTOP-GJL8FJO\noura ... LibraryDataBase 00:00:00 15 rows

Ln 1 Col 1 INS

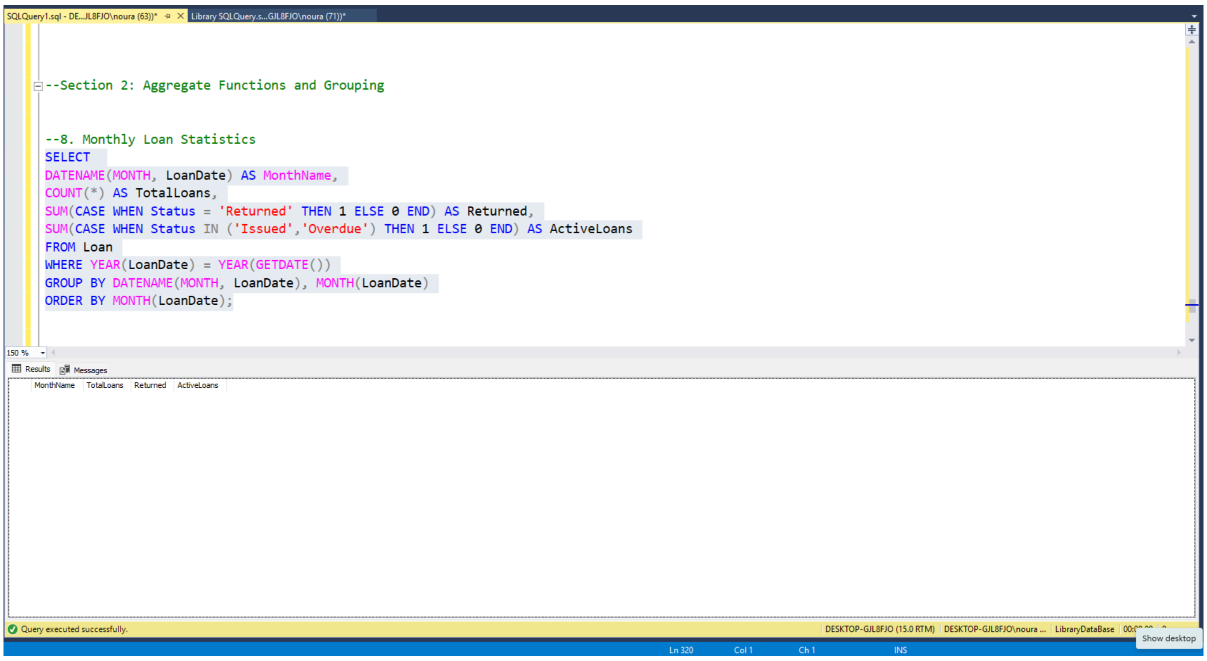


```sql
--7. Revenue Analysis by Genre

SELECT b.Genre,
COUNT(ln.LoanID) TotalLoans,
SUM(p.Amount) TotalRevenue,
AVG(p.Amount) AvgFine
FROM Book b
JOIN Loan ln ON b.BookID=ln.BookID
JOIN Payment p ON ln.LoanID=p.LoanID
GROUP BY b.Genre;
```

	Genre	TotalLoans	TotalRevenue	AvgFine
1	Fiction	1	6.00	6.000000
2	Non-fiction	1	8.00	8.000000
3	Reference	1	4.00	4.000000

Query executed successfully. | DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 3 rows

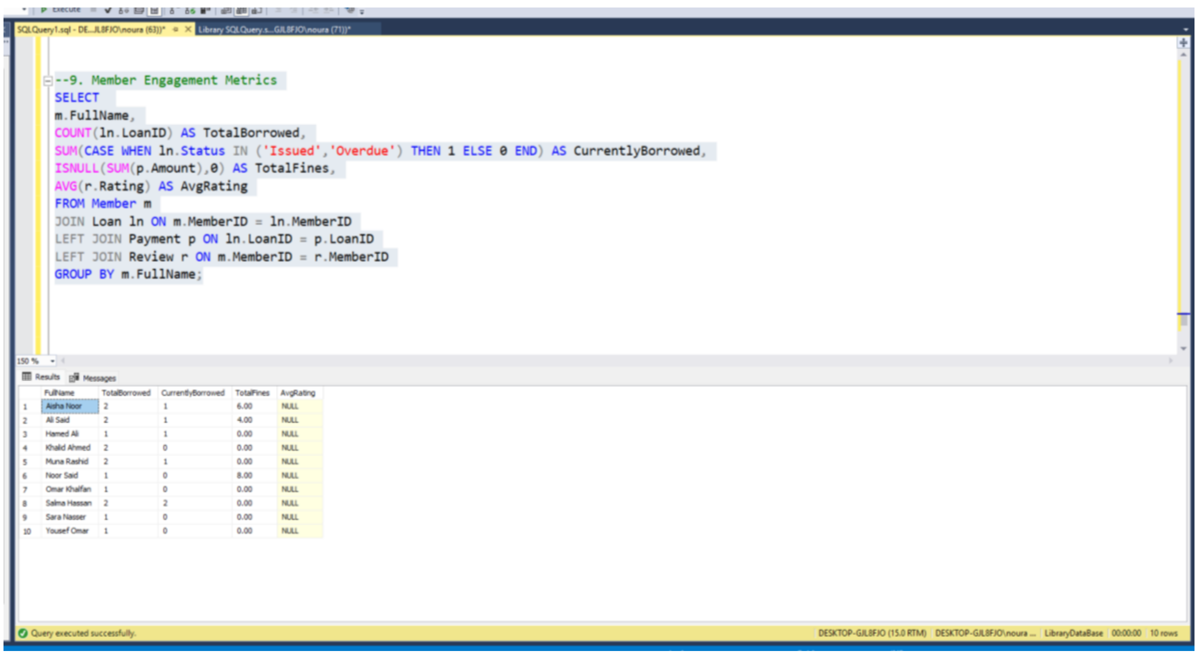
Section 2: Aggregate Functions and Grouping



```sql
--Section 2: Aggregate Functions and Grouping

--8. Monthly Loan Statistics
SELECT
DATENAME(MONTH, LoanDate) AS MonthName,
COUNT(*) AS TotalLoans,
SUM(CASE WHEN Status = 'Returned' THEN 1 ELSE 0 END) AS Returned,
SUM(CASE WHEN Status IN ('Issued','Overdue') THEN 1 ELSE 0 END) AS ActiveLoans
FROM Loan
WHERE YEAR(LoanDate) = YEAR(GETDATE())
GROUP BY DATENAME(MONTH, LoanDate), MONTH(LoanDate)
ORDER BY MONTH(LoanDate);
```

```
MonthName  TotalLoans  Returned  ActiveLoans
```

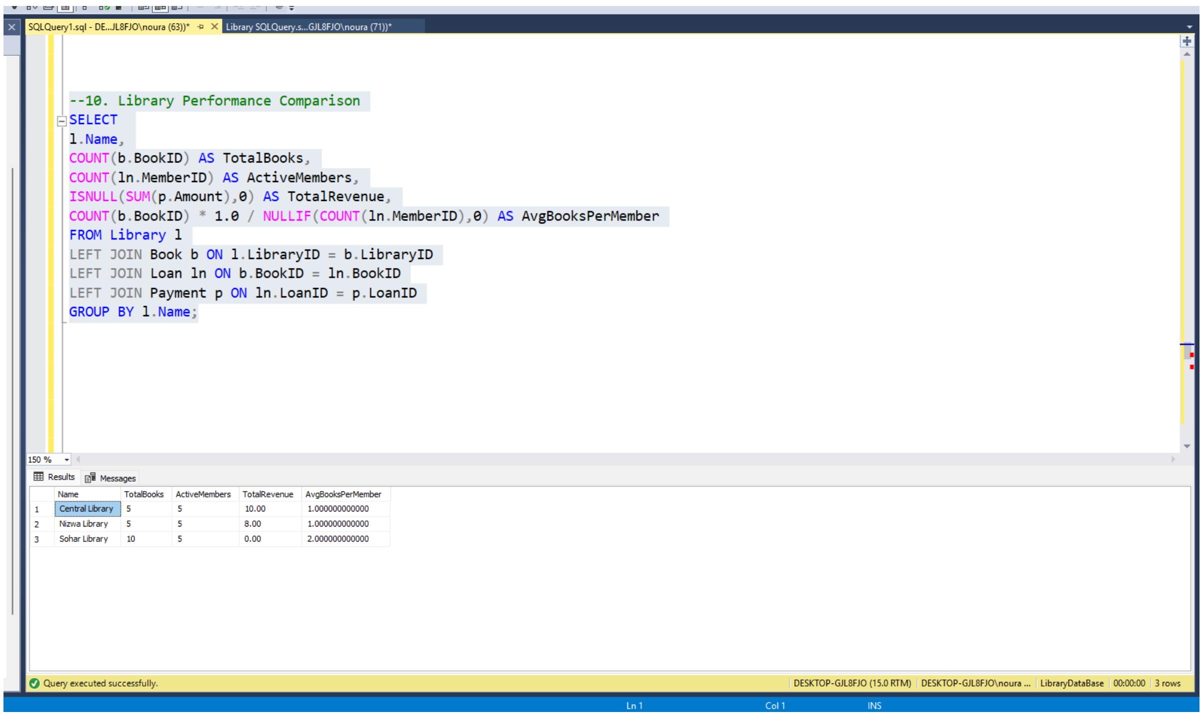
Query executed successfully.



```sql
--9. Member Engagement Metrics
SELECT
m.FullName,
COUNT(ln.LoanID) AS TotalBorrowed,
SUM(CASE WHEN ln.Status IN ('Issued','Overdue') THEN 1 ELSE 0 END) AS CurrentlyBorrowed,
ISNULL(SUM(p.Amount),0) AS TotalFines,
AVG(r.Rating) AS AvgRating
FROM Member m
JOIN Loan ln ON m.MemberID = ln.MemberID
LEFT JOIN Payment p ON ln.LoanID = p.LoanID
LEFT JOIN Review r ON m.MemberID = r.MemberID
GROUP BY m.FullName;
```

```
    FullName       TotalBorrowed  CurrentlyBorrowed  TotalFines  AvgRating
1   Aisha Noor     2              1                  6.00        NULL
2   Ali Said       2              1                  4.00        NULL
3   Hamed Ali      1              1                  0.00        NULL
4   Khalid Ahmed   2              0                  0.00        NULL
5   Muna Rashid    2              1                  0.00        NULL
6   Noor Said      1              0                  8.00        NULL
7   Omar Khalfan   1              0                  0.00        NULL
8   Salma Hassan   2              2                  0.00        NULL
9   Sara Nasser    1              0                  0.00        NULL
10  Yousef Omar    1              0                  0.00        NULL
```

Query executed successfully.

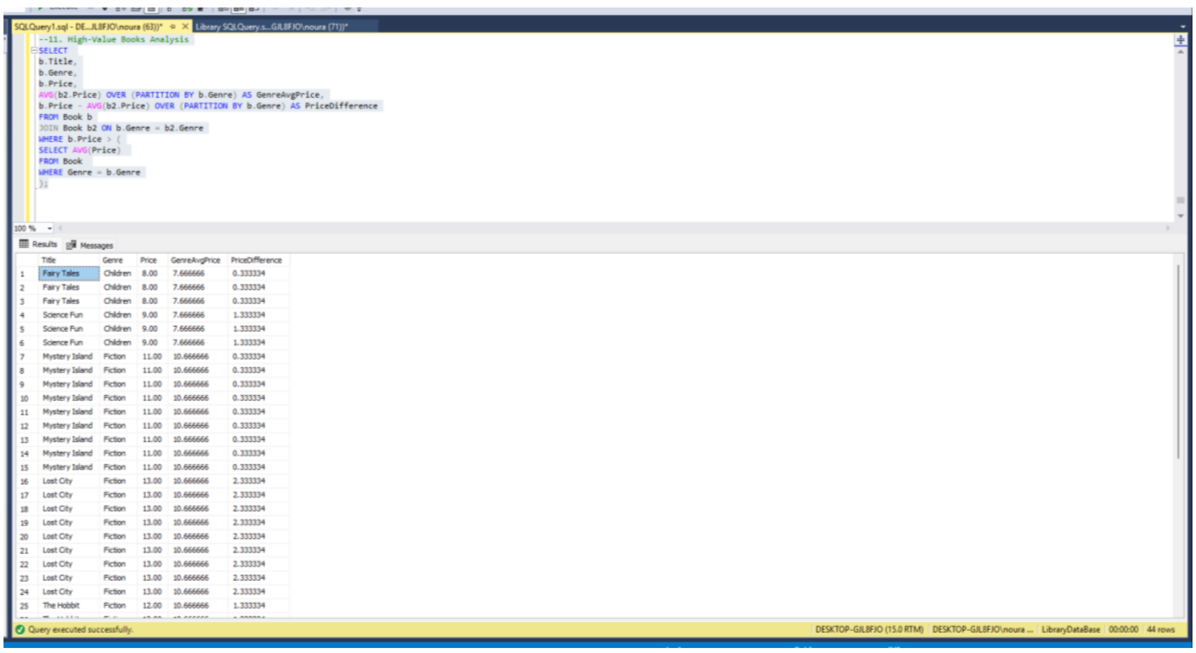


SQLQuery1.sql - DE...JL8FJO\noura (63))*   Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--10. Library Performance Comparison
SELECT
l.Name,
COUNT(b.BookID) AS TotalBooks,
COUNT(ln.MemberID) AS ActiveMembers,
ISNULL(SUM(p.Amount),0) AS TotalRevenue,
COUNT(b.BookID) * 1.0 / NULLIF(COUNT(ln.MemberID),0) AS AvgBooksPerMember
FROM Library l
LEFT JOIN Book b ON l.LibraryID = b.LibraryID
LEFT JOIN Loan ln ON b.BookID = ln.BookID
LEFT JOIN Payment p ON ln.LoanID = p.LoanID
GROUP BY l.Name;
```

	Name	TotalBooks	ActiveMembers	TotalRevenue	AvgBooksPerMember
1	Central Library	5	5	10.00	1.000000000000
2	Nizwa Library	5	5	8.00	1.000000000000
3	Sohar Library	10	5	0.00	2.000000000000

Query executed successfully.   DESKTOP-GJL8FJO (15.0 RTM)   DESKTOP-GJL8FJO\noura ...   LibraryDataBase   00:00:00   3 rows

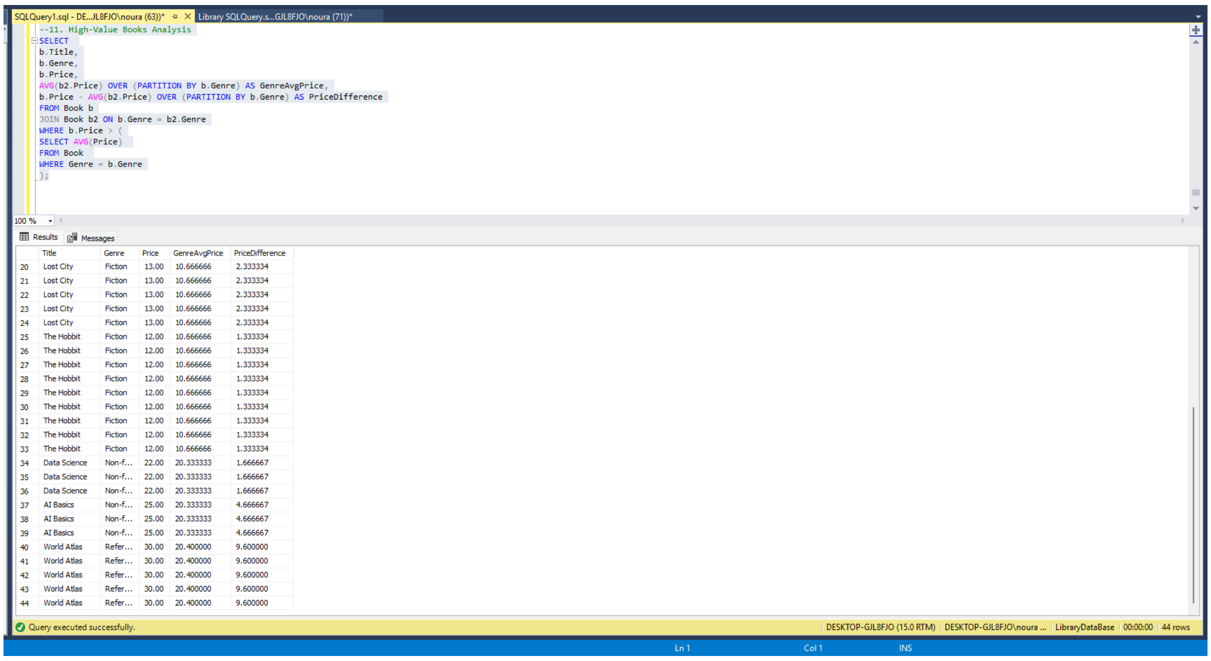


SQLQuery1.sql - DE...JL8FJO\noura (63))*   Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--11. High-Value Books Analysis
SELECT
b.Title,
b.Genre,
b.Price,
AVG(b2.Price) OVER (PARTITION BY b.Genre) AS GenreAvgPrice,
b.Price - AVG(b2.Price) OVER (PARTITION BY b.Genre) AS PriceDifference
FROM Book b
JOIN Book b2 ON b.Genre = b2.Genre
WHERE b.Price > (
SELECT AVG(Price)
FROM Book
WHERE Genre = b.Genre
);
```

	Title	Genre	Price	GenreAvgPrice	PriceDifference
1	Fairy Tales	Children	8.00	7.666666	0.333334
2	Fairy Tales	Children	8.00	7.666666	0.333334
3	Fairy Tales	Children	8.00	7.666666	0.333334
4	Science Fun	Children	9.00	7.666666	1.333334
5	Science Fun	Children	9.00	7.666666	1.333334
6	Science Fun	Children	9.00	7.666666	1.333334
7	Mystery Island	Fiction	11.00	10.666666	0.333334
8	Mystery Island	Fiction	11.00	10.666666	0.333334
9	Mystery Island	Fiction	11.00	10.666666	0.333334
10	Mystery Island	Fiction	11.00	10.666666	0.333334
11	Mystery Island	Fiction	11.00	10.666666	0.333334
12	Mystery Island	Fiction	11.00	10.666666	0.333334
13	Mystery Island	Fiction	11.00	10.666666	0.333334
14	Mystery Island	Fiction	11.00	10.666666	0.333334
15	Mystery Island	Fiction	11.00	10.666666	0.333334
16	Lost City	Fiction	13.00	10.666666	2.333334
17	Lost City	Fiction	13.00	10.666666	2.333334
18	Lost City	Fiction	13.00	10.666666	2.333334
19	Lost City	Fiction	13.00	10.666666	2.333334
20	Lost City	Fiction	13.00	10.666666	2.333334
21	Lost City	Fiction	13.00	10.666666	2.333334
22	Lost City	Fiction	13.00	10.666666	2.333334
23	Lost City	Fiction	13.00	10.666666	2.333334
24	Lost City	Fiction	13.00	10.666666	2.333334
25	The Hobbit	Fiction	12.00	10.666666	1.333334

Query executed successfully.   DESKTOP-GJL8FJO (15.0 RTM)   DESKTOP-GJL8FJO\noura ...   LibraryDataBase   00:00:00   44 rows

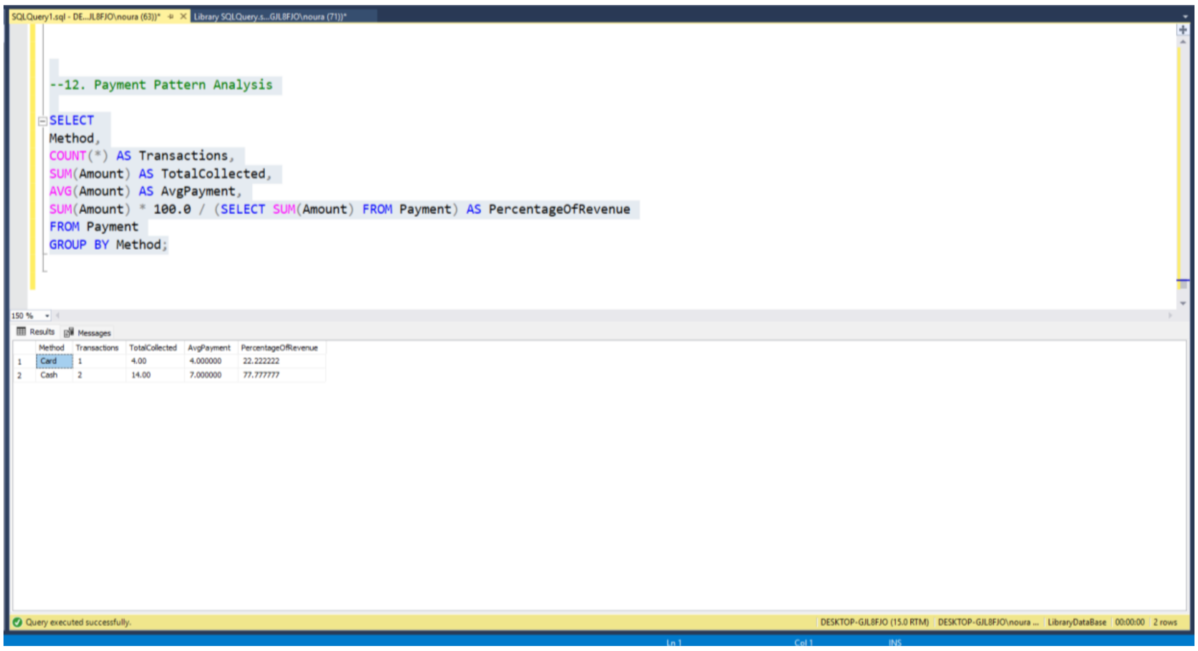


```
--11. High-Value Books Analysis
SELECT
b.Title,
b.Genre,
b.Price,
AVG(b2.Price) OVER (PARTITION BY b.Genre) AS GenreAvgPrice,
b.Price - AVG(b2.Price) OVER (PARTITION BY b.Genre) AS PriceDifference
FROM Book b
JOIN Book b2 ON b.Genre = b2.Genre
WHERE b.Price > (
SELECT AVG(Price)
FROM Book
WHERE Genre = b.Genre
);
```

	Title	Genre	Price	GenreAvgPrice	PriceDifference
20	Lost City	Fiction	13.00	10.666666	2.333334
21	Lost City	Fiction	13.00	10.666666	2.333334
22	Lost City	Fiction	13.00	10.666666	2.333334
23	Lost City	Fiction	13.00	10.666666	2.333334
24	Lost City	Fiction	13.00	10.666666	2.333334
25	The Hobbit	Fiction	12.00	10.666666	1.333334
26	The Hobbit	Fiction	12.00	10.666666	1.333334
27	The Hobbit	Fiction	12.00	10.666666	1.333334
28	The Hobbit	Fiction	12.00	10.666666	1.333334
29	The Hobbit	Fiction	12.00	10.666666	1.333334
30	The Hobbit	Fiction	12.00	10.666666	1.333334
31	The Hobbit	Fiction	12.00	10.666666	1.333334
32	The Hobbit	Fiction	12.00	10.666666	1.333334
33	The Hobbit	Fiction	12.00	10.666666	1.333334
34	Data Science	Non-f...	22.00	20.333333	1.666667
35	Data Science	Non-f...	22.00	20.333333	1.666667
36	Data Science	Non-f...	22.00	20.333333	1.666667
37	AI Basics	Non-f...	25.00	20.333333	4.666667
38	AI Basics	Non-f...	25.00	20.333333	4.666667
39	AI Basics	Non-f...	25.00	20.333333	4.666667
40	World Atlas	Refer...	30.00	20.400000	9.600000
41	World Atlas	Refer...	30.00	20.400000	9.600000
42	World Atlas	Refer...	30.00	20.400000	9.600000
43	World Atlas	Refer...	30.00	20.400000	9.600000
44	World Atlas	Refer...	30.00	20.400000	9.600000

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 44 rows
```

SQLQuery1.sql - DE...JL8FJO\noura (63))*
Library SQLQuery.s...GJL8FJO\noura (71))*
100 %
Results
Messages
Ln 1
Col 1
INS



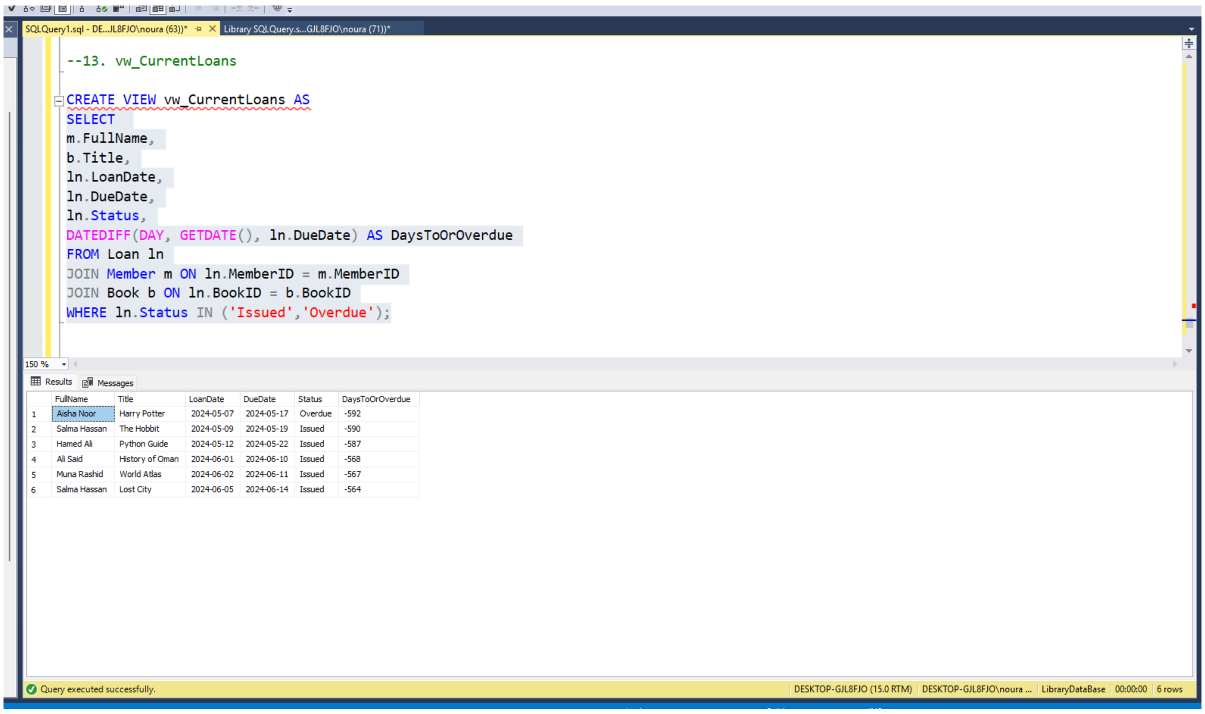
```
--12. Payment Pattern Analysis

SELECT
Method,
COUNT(*) AS Transactions,
SUM(Amount) AS TotalCollected,
AVG(Amount) AS AvgPayment,
SUM(Amount) * 100.0 / (SELECT SUM(Amount) FROM Payment) AS PercentageOfRevenue
FROM Payment
GROUP BY Method;
```

	Method	Transactions	TotalCollected	AvgPayment	PercentageOfRevenue
1	Card	1	4.00	4.000000	22.222222
2	Cash	2	14.00	7.000000	77.777777

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 2 rows
```

SQLQuery1.sql - DE...JL8FJO\noura (63))*
Library SQLQuery.s...GJL8FJO\noura (71))*
150 %
Results
Messages
Ln 1
Col 1
INS

Section 3: Views Creation



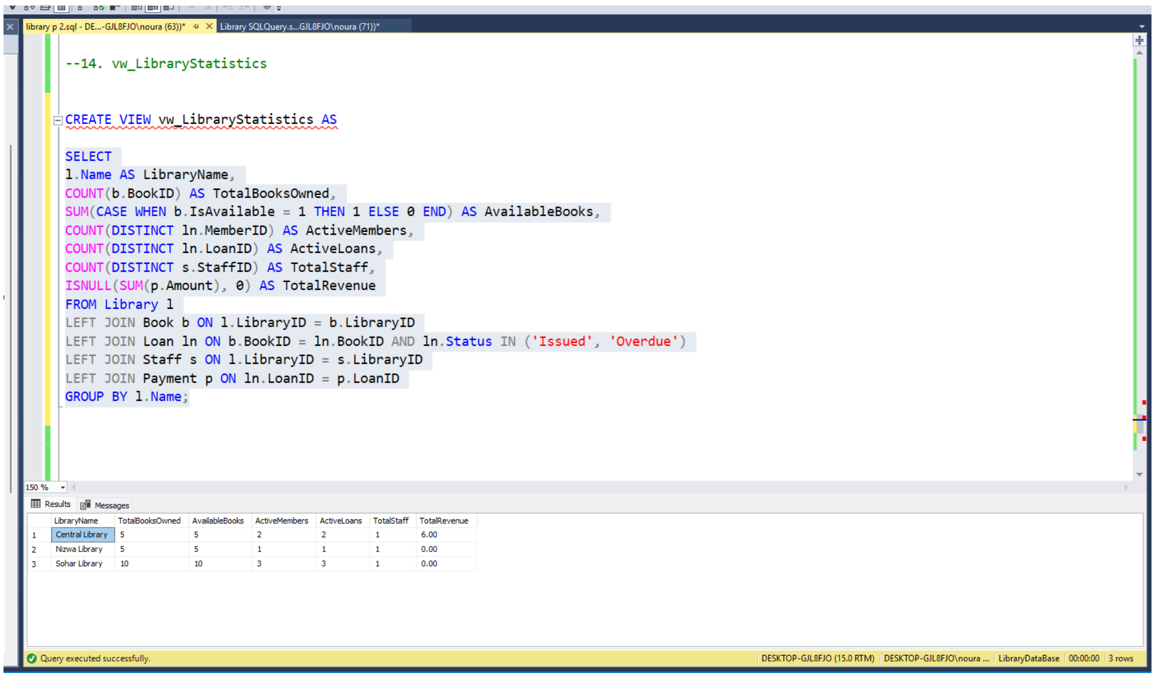
SQLQuery1.sql - DE...JL8FJO\noura (63))* | Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--13. vw_CurrentLoans

CREATE VIEW vw_CurrentLoans AS
SELECT
m.FullName,
b.Title,
ln.LoanDate,
ln.DueDate,
ln.Status,
DATEDIFF(DAY, GETDATE(), ln.DueDate) AS DaysToOrOverdue
FROM Loan ln
JOIN Member m ON ln.MemberID = m.MemberID
JOIN Book b ON ln.BookID = b.BookID
WHERE ln.Status IN ('Issued','Overdue');
```

	FullName	Title	LoanDate	DueDate	Status	DaysToOrOverdue
1	Aisha Noor	Harry Potter	2024-05-07	2024-05-17	Overdue	-592
2	Salma Hassan	The Hobbit	2024-05-09	2024-05-19	Issued	-590
3	Hamed Ali	Python Guide	2024-05-12	2024-05-22	Issued	-587
4	Ali Said	History of Oman	2024-06-01	2024-06-10	Issued	-568
5	Muna Rashid	World Atlas	2024-06-02	2024-06-11	Issued	-567
6	Salma Hassan	Lost City	2024-06-05	2024-06-14	Issued	-564

Query executed successfully. | DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 6 rows

Q14- vw_LibraryStatistics should show library name, total books owned by the library, number of available books, total active members(members who have at least one loan from this library's books), active loans (loans of books belonging to this library), total staff working at the library, total revenue from fines (from loans of this library's books).

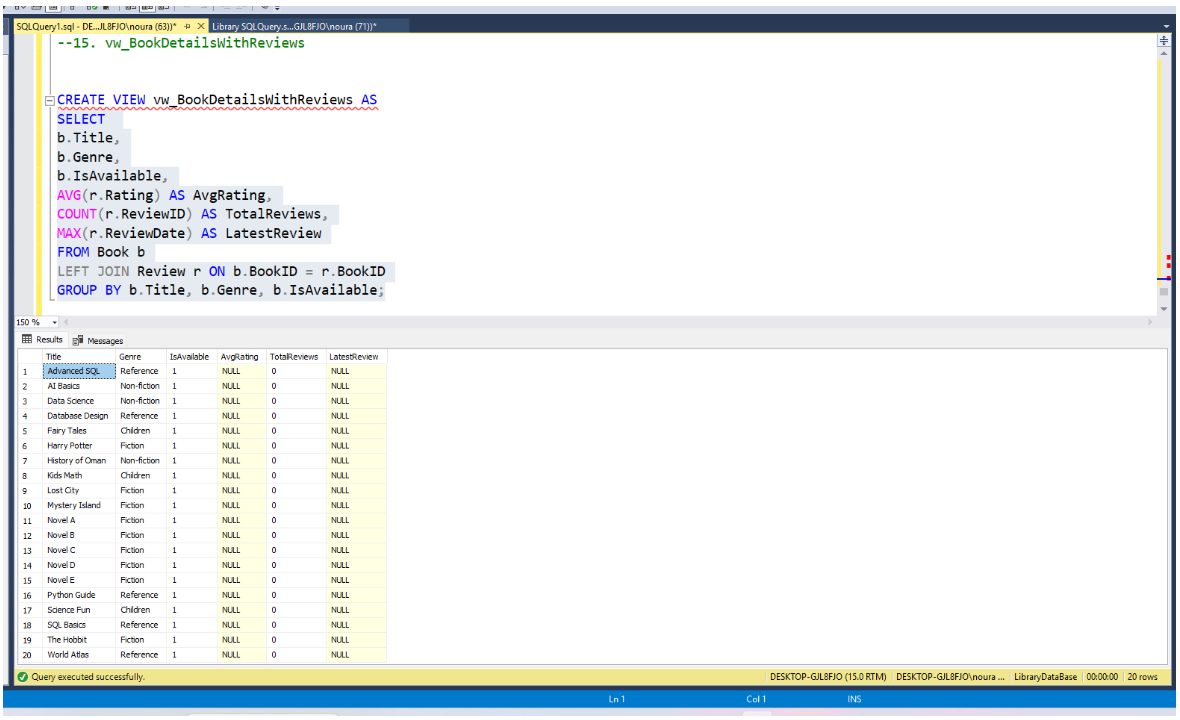


library p 2.sql - DE...-GJL8FJO\noura (63))* | Library SQLQuery.s...GJL8FJO\noura (71))*

```sql
--14. vw_LibraryStatistics

CREATE VIEW vw_LibraryStatistics AS

SELECT
l.Name AS LibraryName,
COUNT(b.BookID) AS TotalBooksOwned,
SUM(CASE WHEN b.IsAvailable = 1 THEN 1 ELSE 0 END) AS AvailableBooks,
COUNT(DISTINCT ln.MemberID) AS ActiveMembers,
COUNT(DISTINCT ln.LoanID) AS ActiveLoans,
COUNT(DISTINCT s.StaffID) AS TotalStaff,
ISNULL(SUM(p.Amount), 0) AS TotalRevenue
FROM Library l
LEFT JOIN Book b ON l.LibraryID = b.LibraryID
LEFT JOIN Loan ln ON b.BookID = ln.BookID AND ln.Status IN ('Issued', 'Overdue')
LEFT JOIN Staff s ON l.LibraryID = s.LibraryID
LEFT JOIN Payment p ON ln.LoanID = p.LoanID
GROUP BY l.Name;
```

	LibraryName	TotalBooksOwned	AvailableBooks	ActiveMembers	ActiveLoans	TotalStaff	TotalRevenue
1	Central Library	5	5	2	2	1	6.00
2	Nizwa Library	5	5	1	1	1	0.00
3	Sohar Library	10	10	3	3	1	0.00

Query executed successfully. | DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 3 rows



```sql
--15. vw_BookDetailsWithReviews


CREATE VIEW vw_BookDetailsWithReviews AS
SELECT
b.Title,
b.Genre,
b.IsAvailable,
AVG(r.Rating) AS AvgRating,
COUNT(r.ReviewID) AS TotalReviews,
MAX(r.ReviewDate) AS LatestReview
FROM Book b
LEFT JOIN Review r ON b.BookID = r.BookID
GROUP BY b.Title, b.Genre, b.IsAvailable;
```

	Title	Genre	IsAvailable	AvgRating	TotalReviews	LatestReview
1	Advanced SQL	Reference	1	NULL	0	NULL
2	AI Basics	Non-fiction	1	NULL	0	NULL
3	Data Science	Non-fiction	1	NULL	0	NULL
4	Database Design	Reference	1	NULL	0	NULL
5	Fairy Tales	Children	1	NULL	0	NULL
6	Harry Potter	Fiction	1	NULL	0	NULL
7	History of Oman	Non-fiction	1	NULL	0	NULL
8	Kids Math	Children	1	NULL	0	NULL
9	Lost City	Fiction	1	NULL	0	NULL
10	Mystery Island	Fiction	1	NULL	0	NULL
11	Novel A	Fiction	1	NULL	0	NULL
12	Novel B	Fiction	1	NULL	0	NULL
13	Novel C	Fiction	1	NULL	0	NULL
14	Novel D	Fiction	1	NULL	0	NULL
15	Novel E	Fiction	1	NULL	0	NULL
16	Python Guide	Reference	1	NULL	0	NULL
17	Science Fun	Children	1	NULL	0	NULL
18	SQL Basics	Reference	1	NULL	0	NULL
19	The Hobbit	Fiction	1	NULL	0	NULL
20	World Atlas	Reference	1	NULL	0	NULL

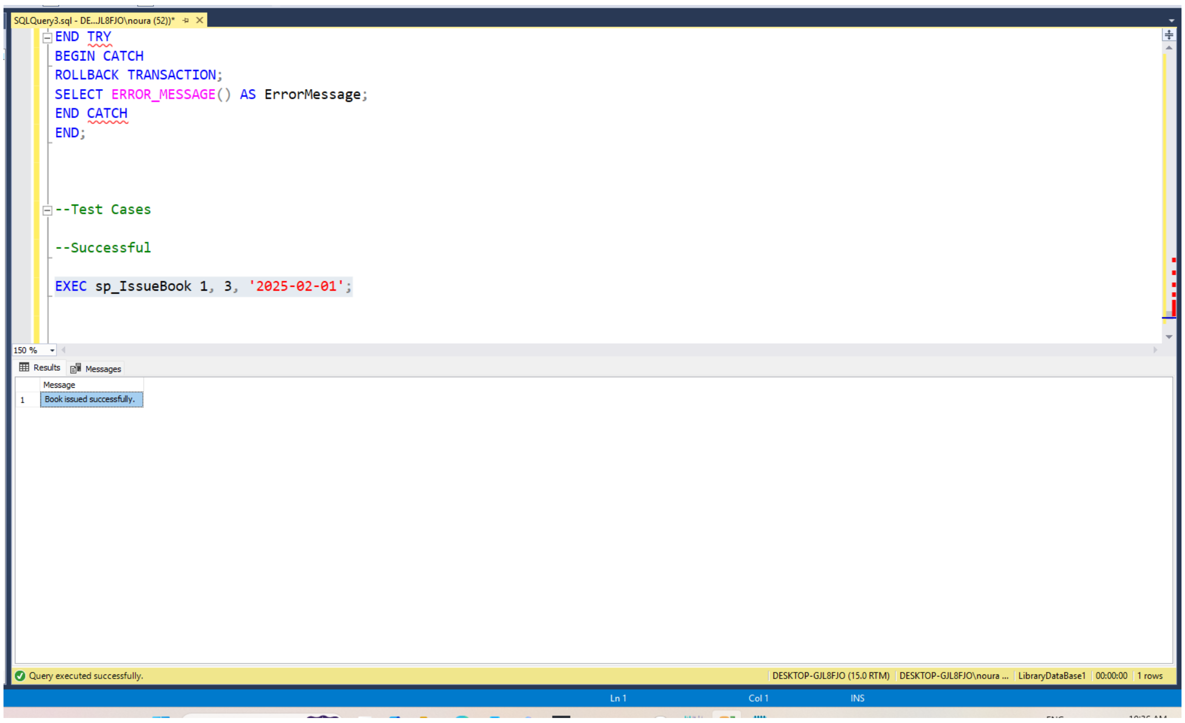
```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase | 00:00:00 | 20 rows
```

Section 4: Stored Procedures+ Testing Evidence

Testing Evidence

16. sp_IssueBook

- **One successful execution**

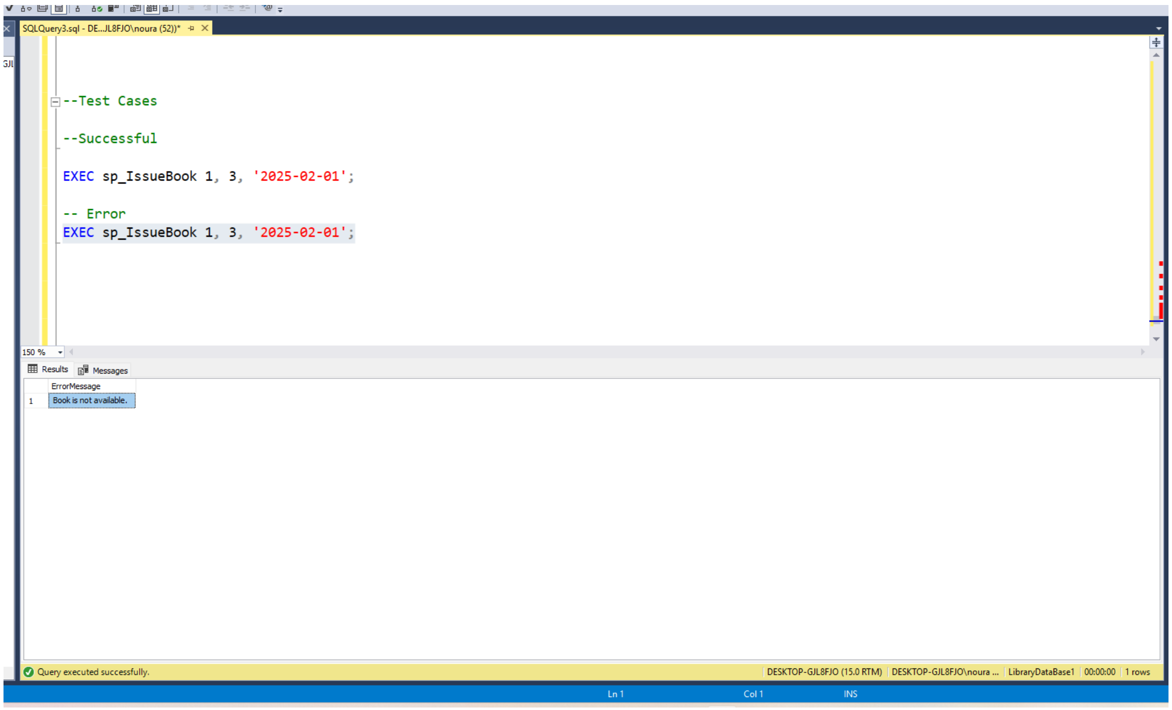


```sql
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION;
SELECT ERROR_MESSAGE() AS ErrorMessage;
END CATCH
END;



--Test Cases

--Successful

EXEC sp_IssueBook 1, 3, '2025-02-01';
```

	Message
1	Book issued successfully.

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 1 rows
```

• **One that demonstrates error handling or validation**



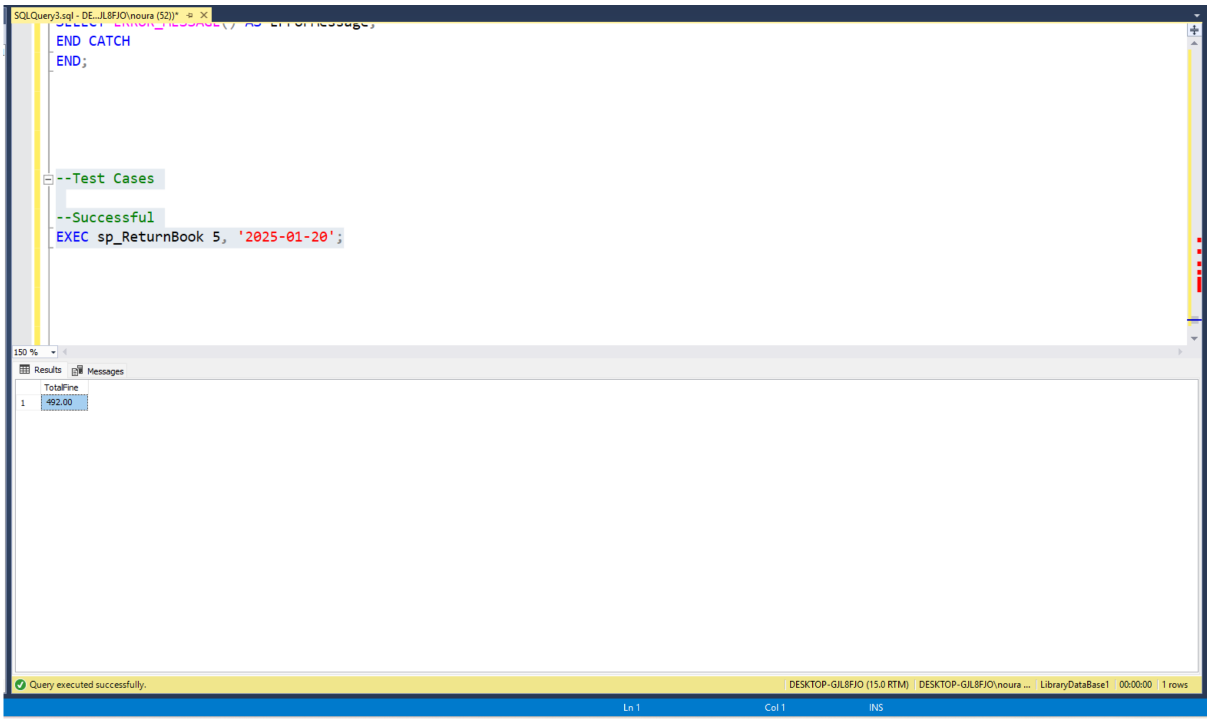
```
--Test Cases

--Successful

EXEC sp_IssueBook 1, 3, '2025-02-01';

-- Error
EXEC sp_IssueBook 1, 3, '2025-02-01';
```

```
   ErrorMessage
1  Book is not available.
```

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 1 rows
```

17. sp_ReturnBook

• **One successful execution**

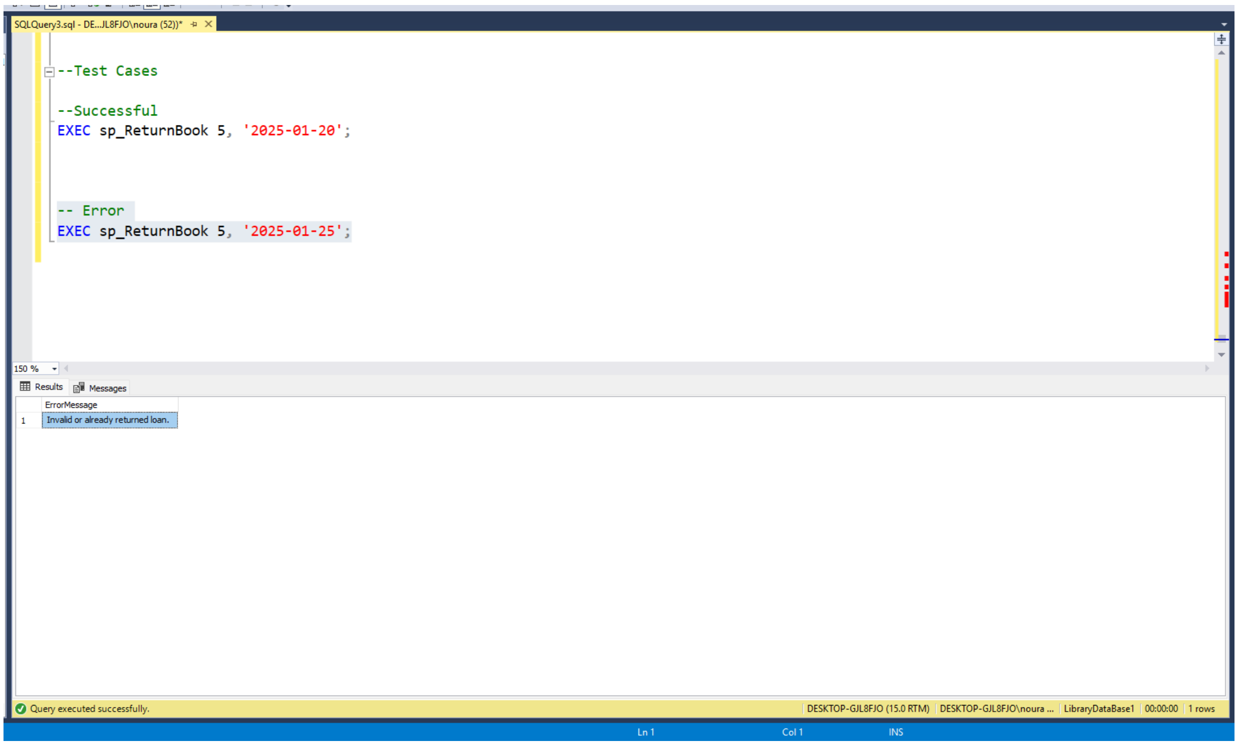


```
END CATCH
END;



--Test Cases

--Successful
EXEC sp_ReturnBook 5, '2025-01-20';
```

```
   TotalFine
1  492.00
```

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 1 rows
```

- **One that demonstrates error handling or validation**



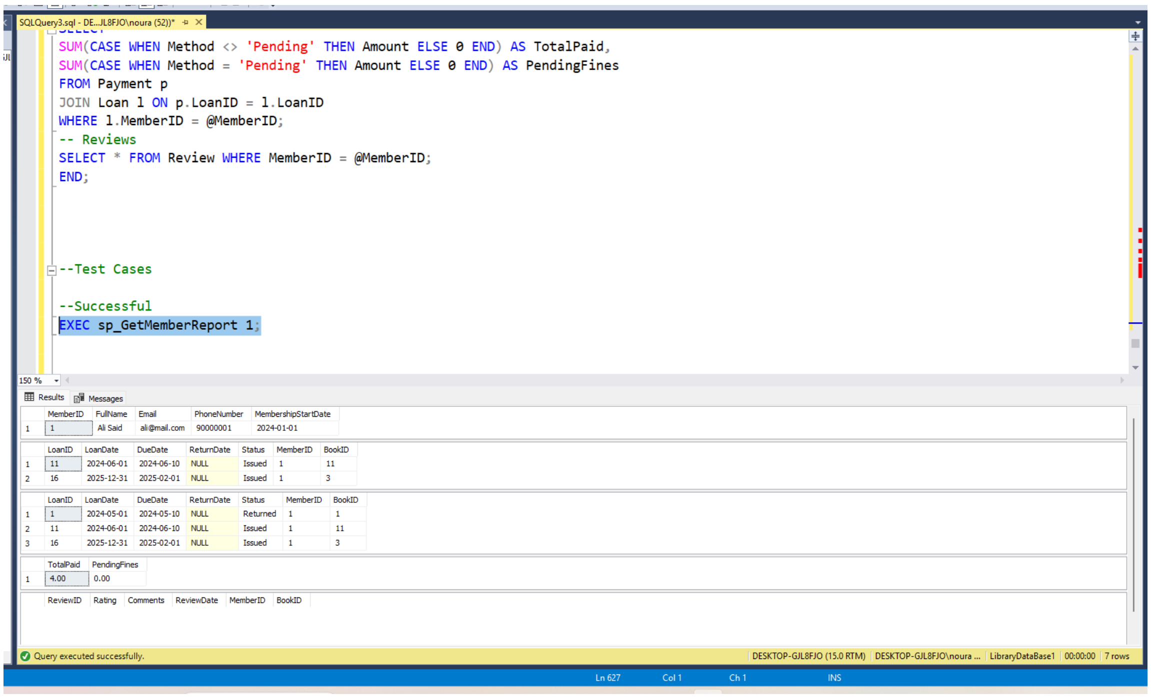
```sql
--Test Cases

--Successful
EXEC sp_ReturnBook 5, '2025-01-20';


-- Error
EXEC sp_ReturnBook 5, '2025-01-25';
```

	ErrorMessage
1	Invalid or already returned loan.

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 1 rows
Ln 1    Col 1    INS
```

18. sp_GetMemberReport

- **One successful execution**



```sql
SUM(CASE WHEN Method <> 'Pending' THEN Amount ELSE 0 END) AS TotalPaid,
SUM(CASE WHEN Method = 'Pending' THEN Amount ELSE 0 END) AS PendingFines
FROM Payment p
JOIN Loan l ON p.LoanID = l.LoanID
WHERE l.MemberID = @MemberID;
-- Reviews
SELECT * FROM Review WHERE MemberID = @MemberID;
END;



--Test Cases

--Successful
EXEC sp_GetMemberReport 1;
```

	MemberID	FullName	Email	PhoneNumber	MembershipStartDate
1	1	Ali Said	ali@mail.com	90000001	2024-01-01

	LoanID	LoanDate	DueDate	ReturnDate	Status	MemberID	BookID
1	11	2024-06-01	2024-06-10	NULL	Issued	1	11
2	16	2025-12-31	2025-02-01	NULL	Issued	1	3

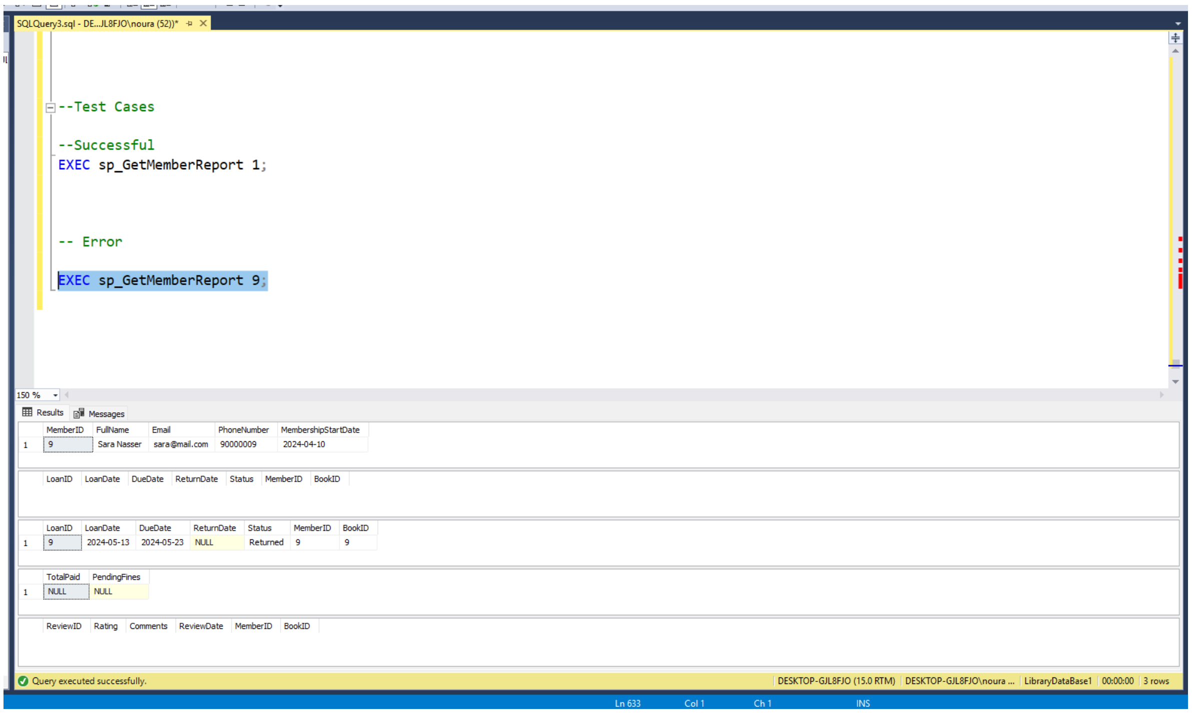
	LoanID	LoanDate	DueDate	ReturnDate	Status	MemberID	BookID
1	1	2024-05-01	2024-05-10	NULL	Returned	1	1
2	11	2024-06-01	2024-06-10	NULL	Issued	1	11
3	16	2025-12-31	2025-02-01	NULL	Issued	1	3

	TotalPaid	PendingFines
1	4.00	0.00

	ReviewID	Rating	Comments	ReviewDate	MemberID	BookID

```
Query executed successfully.    DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 7 rows
Ln 627    Col 1    Ch 1    INS
```

• **One that demonstrates error handling or validation**



SQLQuery3.sql - DE...JL8FJO\noura (52))*

```sql
--Test Cases

--Successful
EXEC sp_GetMemberReport 1;


-- Error

EXEC sp_GetMemberReport 9;
```

150 %

Results | Messages

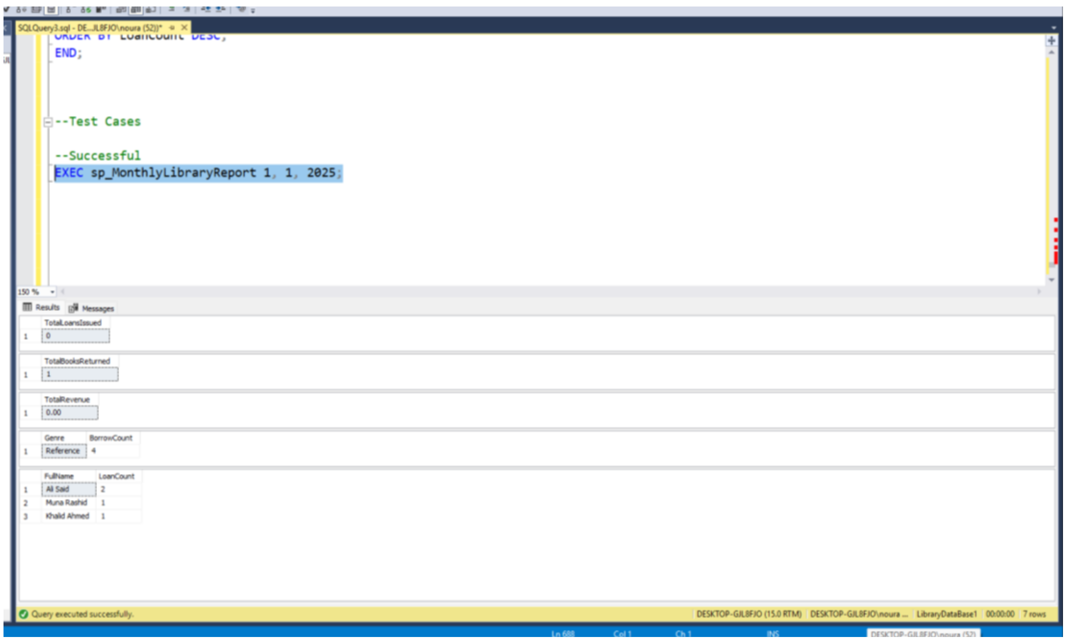
```
    MemberID  FullName     Email           PhoneNumber  MembershipStartDate
1   9         Sara Nasser  sara@mail.com   90000009     2024-04-10

    LoanID  LoanDate  DueDate  ReturnDate  Status  MemberID  BookID

    LoanID  LoanDate    DueDate     ReturnDate  Status    MemberID  BookID
1   9       2024-05-13  2024-05-23  NULL        Returned  9         9

    TotalPaid  PendingFines
1   NULL       NULL

    ReviewID  Rating  Comments  ReviewDate  MemberID  BookID
```

Query executed successfully. | DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 3 rows

Ln 633 Col 1 Ch 1 INS

19. sp_MonthlyLibraryReport

• **One successful execution**



SQLQuery3.sql - DE...JL8FJO\noura (52))*

```sql
ORDER BY LoanCount DESC;
END;


--Test Cases

--Successful
EXEC sp_MonthlyLibraryReport 1, 1, 2025;
```

150 %

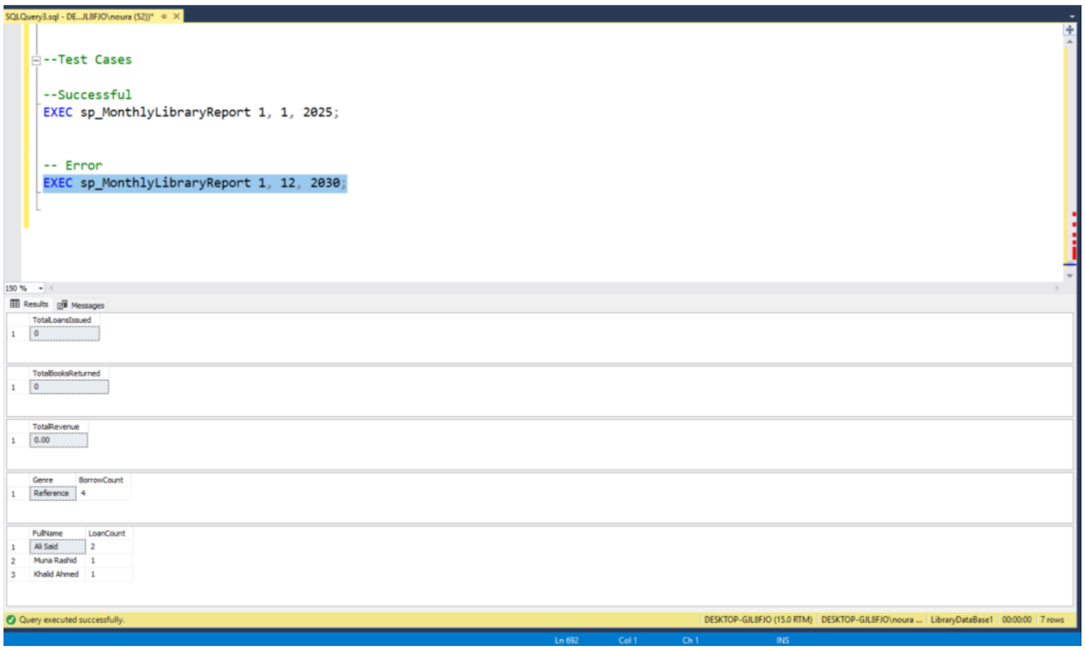
Results | Messages

```
    TotalLoansIssued
1   0

    TotalBooksReturned
1   1

    TotalRevenue
1   0.00

    Genre      BorrowCount
1   Reference  4

    FullName      LoanCount
1   Ali Said      2
2   Muna Rashid   1
3   Khalid Ahmed  1
```

Query executed successfully. | DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 7 rows

Ln 688 Col 1 Ch 1 INS

• **One that demonstrates error handling or validation** (Month with no activity)



```
--Test Cases

--Successful
EXEC sp_MonthlyLibraryReport 1, 1, 2025;


-- Error
EXEC sp_MonthlyLibraryReport 1, 12, 2030;
```

```
   TotalLoansIssued
1  0

   TotalBooksReturned
1  0

   TotalRevenue
1  0.00

   Genre      BorrowCount
1  Reference  4

   FullName      LoanCount
1  Ali Said      2
2  Muna Rashid   1
3  Khalid Ahmed  1

Query executed successfully.   DESKTOP-GJL8FJO (15.0 RTM) | DESKTOP-GJL8FJO\noura ... | LibraryDataBase1 | 00:00:00 | 7 rows
```

• Explanation of how your stored procedures handle edge cases

Q16 – `sp_IssueBook` (Issue Book Procedure)

Edge Case 1: Book is not available

- The procedure checks `Book.IsAvailable = 1` before issuing.
- If the book is already issued, the procedure **throws an error** and **rolls back the transaction**.
- This prevents multiple members from borrowing the same book.

Edge Case 2: Member has overdue loans

- The procedure checks the `Loan` table for any `Status = 'Overdue'` loans for the member.
- If an overdue loan exists, the issue request is rejected.
- This enforces library borrowing rules.

Edge Case 3: Partial updates (data inconsistency)

- Loan insertion and book availability update are wrapped in a **transaction**.
- If any validation fails, **ROLLBACK** ensures no changes are saved.

Q17 – `sp_ReturnBook` (Return Book Procedure)

Edge Case 1: Loan does not exist or already returned

- The procedure validates that the loan exists and is not already marked as `Returned`.
- If invalid, an error is raised and no changes are made.

Edge Case 2: Late return (overdue fine calculation)

- The procedure uses `DATEDIFF` to calculate overdue days.
- A fine of **$2 per day overdue** is automatically calculated.
- Ensures accurate and consistent fine computation.

Edge Case 3: Fine creation and consistency

- If a fine exists, a payment record is automatically created.
- All updates (loan status, book availability, payment record) occur in **one transaction**.

Q18 – `sp_GetMemberReport` (Member Report Procedure)

Edge Case 1: Member has no current loans

- The procedure returns empty result sets without errors.
- This ensures the report still runs successfully.

Edge Case 2: Member has no fines or reviews

- Aggregation queries safely handle `NULL` values.
- The procedure returns `NULL` or zero totals rather than failing.

Edge Case 3: Read-only operation

- No transaction is used because the procedure performs **SELECT operations only**.
- This follows best practices and avoids unnecessary locking.

Q19 – `sp_MonthlyLibraryReport` (Monthly Library Report)

Edge Case 1: Month with no activity

- If no loans, returns, or payments exist for the given month/year, queries return zero or empty results.
- Prevents runtime errors and supports consistent reporting.

Edge Case 2: Library with no revenue

- Revenue calculations use aggregation functions that handle missing payment data.
- The procedure still executes successfully.

Edge Case 3: Multiple result sets consistency

- Each metric is calculated independently, ensuring partial data does not affect other results.

IN SHORT:

Question	Procedure	Key Edge Cases Handled
Q16	`sp_IssueBook`	Book unavailable, overdue member, transaction rollback
Q17	`sp_ReturnBook`	Late return, fine calculation, invalid loan
Q18	`sp_GetMemberReport`	No loans, no fines, no reviews
Q19	`sp_MonthlyLibraryReport`	No activity month, no revenue

• Any assumptions I made

Book Availability Assumption

- A book can be loaned to only one member at a time.

- This is enforced using the `IsAvailable` attribute in the Book table.
- When a book is issued, `IsAvailable` is set to `FALSE`, and when returned, it is set back to TRUE.

Loan Status Management

- Loan status values are limited to Issued, Returned, and Overdue.
- The system assumes:
  - `Issued` → Book currently borrowed
  - `Returned` → Book successfully returned
  - `Overdue` → Due date has passed and book is not returned
- Overdue status is updated logically via queries and procedures, not automatically by triggers.

Fine Calculation Rule

- A fine of $2 per day overdue is applied.
- Fines are calculated only at the time of book return.
- Partial days are not counted (only full overdue days using `DATEDIFF`).

Payment Assumptions

- Each payment is associated with one specific loan.
- Payments represent fine payments only (no membership or purchase fees).
- Payment method values (e.g., Cash, Card) are assumed to be valid without a lookup table.

Member Borrowing Rules

- Members are not allowed to borrow new books if they have any overdue loans.
- Members can borrow multiple books simultaneously, as long as none are overdue.

Review Rules

- Members can submit multiple reviews, even for the same book.
- Reviews are optional and may include no comments, in which case a default value is used.
- Reviews are not restricted to returned books only (assumed allowed).

Library–Staff Relationship

- Each staff member works for only one library.

- Each library must have at least one staff member, but this is enforced by data insertion rules rather than database constraints.

Date and Time Handling

- All date values use SQL Server’s DATE data type.
- The system assumes:
  - Server date (`GETDATE()`) represents the current date accurately.
  - Time components are not required for business logic.

Deletion Behavior

- Cascading deletes are enabled.
- If a parent record (e.g., Member, Book, Library) is deleted:
  - Related loans, payments, and reviews are automatically deleted.
- This assumes deletions are intentional and controlled by administrators.

Reporting Scope

- Reports and analytics are based only on data stored within the system.
- External integrations (e.g., payment gateways or external catalogs) are out of scope.