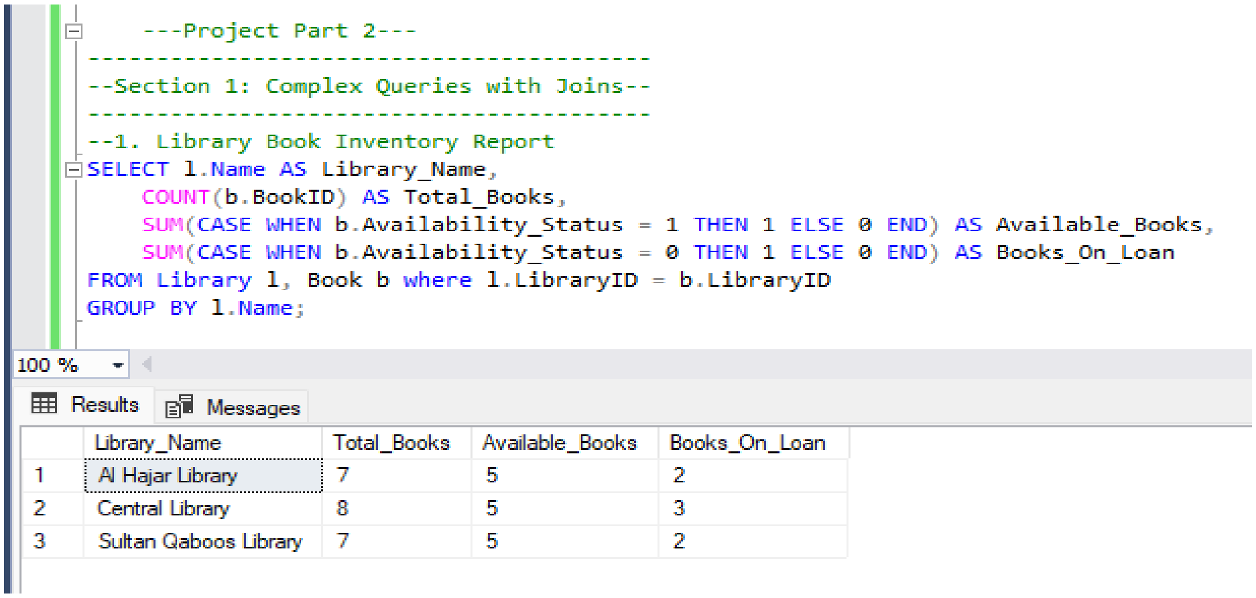
Section 1: Complex Queries with Joins

1. Library Book Inventory Report



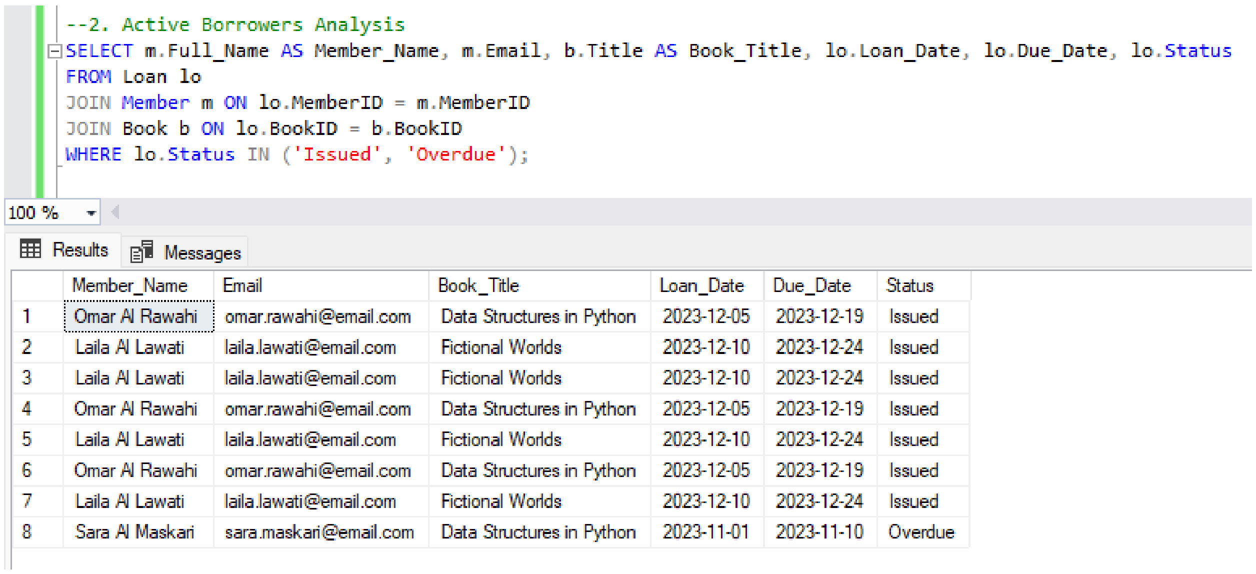
```
   ---Project Part 2---
-----------------------------------------
--Section 1: Complex Queries with Joins--
-----------------------------------------
--1. Library Book Inventory Report
SELECT l.Name AS Library_Name,
    COUNT(b.BookID) AS Total_Books,
    SUM(CASE WHEN b.Availability_Status = 1 THEN 1 ELSE 0 END) AS Available_Books,
    SUM(CASE WHEN b.Availability_Status = 0 THEN 1 ELSE 0 END) AS Books_On_Loan
FROM Library l, Book b where l.LibraryID = b.LibraryID
GROUP BY l.Name;
```

100 %

Results | Messages

	Library_Name	Total_Books	Available_Books	Books_On_Loan
1	Al Hajar Library	7	5	2
2	Central Library	8	5	3
3	Sultan Qaboos Library	7	5	2

2. Active Borrowers Analysis



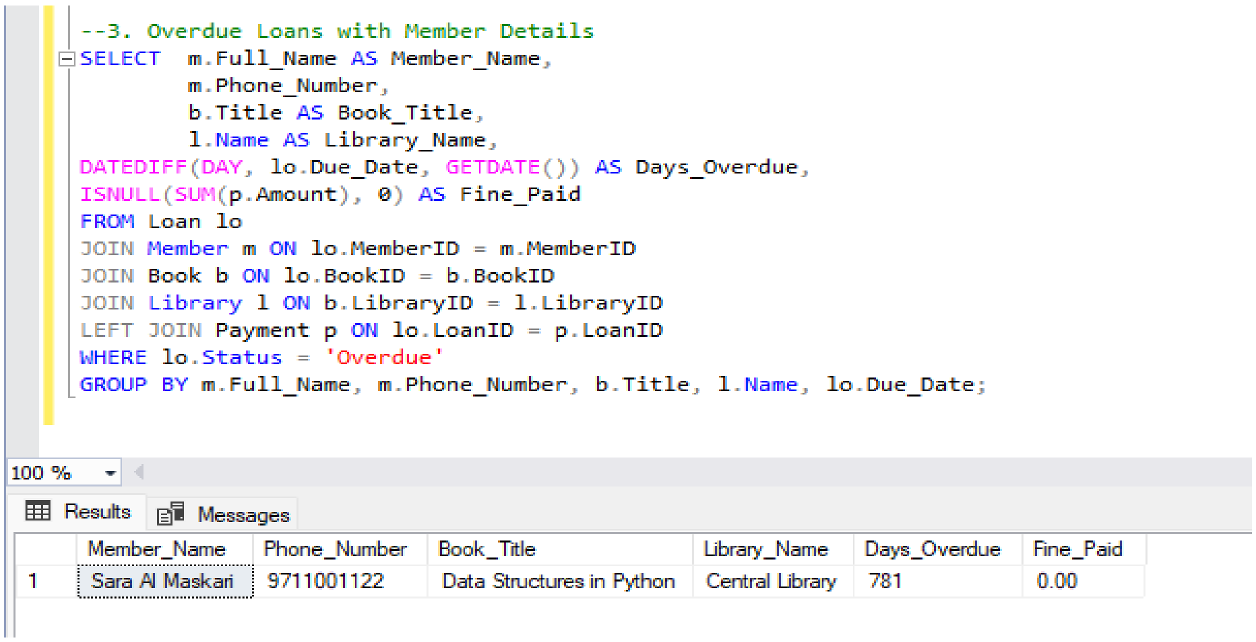
```
--2. Active Borrowers Analysis
SELECT m.Full_Name AS Member_Name, m.Email, b.Title AS Book_Title, lo.Loan_Date, lo.Due_Date, lo.Status
FROM Loan lo
JOIN Member m ON lo.MemberID = m.MemberID
JOIN Book b ON lo.BookID = b.BookID
WHERE lo.Status IN ('Issued', 'Overdue');
```

100 %

Results | Messages

	Member_Name	Email	Book_Title	Loan_Date	Due_Date	Status
1	Omar Al Rawahi	omar.rawahi@email.com	Data Structures in Python	2023-12-05	2023-12-19	Issued
2	Laila Al Lawati	laila.lawati@email.com	Fictional Worlds	2023-12-10	2023-12-24	Issued
3	Laila Al Lawati	laila.lawati@email.com	Fictional Worlds	2023-12-10	2023-12-24	Issued
4	Omar Al Rawahi	omar.rawahi@email.com	Data Structures in Python	2023-12-05	2023-12-19	Issued
5	Laila Al Lawati	laila.lawati@email.com	Fictional Worlds	2023-12-10	2023-12-24	Issued
6	Omar Al Rawahi	omar.rawahi@email.com	Data Structures in Python	2023-12-05	2023-12-19	Issued
7	Laila Al Lawati	laila.lawati@email.com	Fictional Worlds	2023-12-10	2023-12-24	Issued
8	Sara Al Maskari	sara.maskari@email.com	Data Structures in Python	2023-11-01	2023-11-10	Overdue

3. Overdue Loans with Member Details



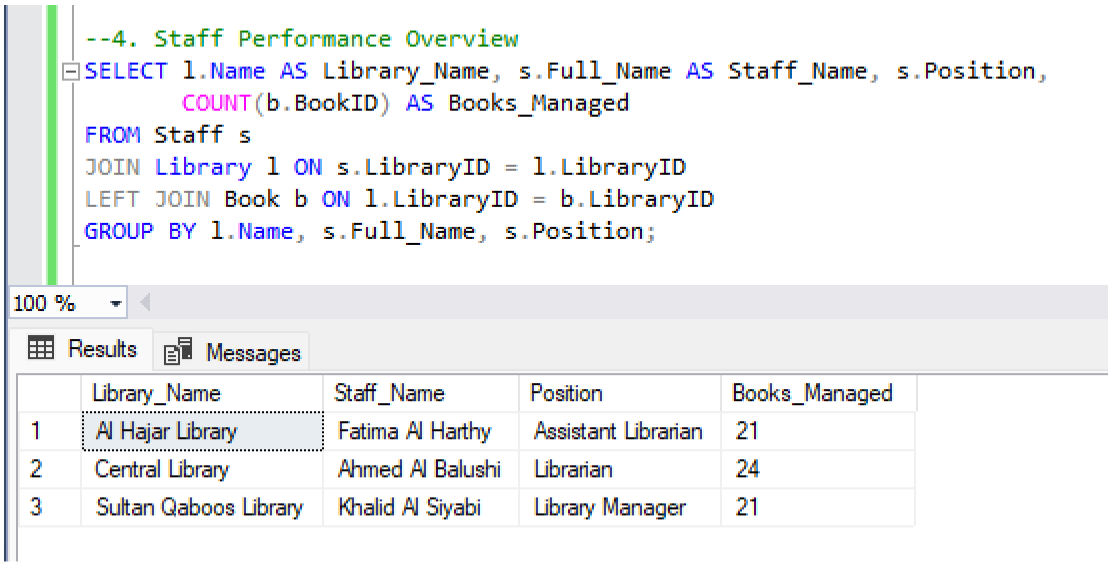
```sql
--3. Overdue Loans with Member Details
SELECT  m.Full_Name AS Member_Name,
        m.Phone_Number,
        b.Title AS Book_Title,
        l.Name AS Library_Name,
DATEDIFF(DAY, lo.Due_Date, GETDATE()) AS Days_Overdue,
ISNULL(SUM(p.Amount), 0) AS Fine_Paid
FROM Loan lo
JOIN Member m ON lo.MemberID = m.MemberID
JOIN Book b ON lo.BookID = b.BookID
JOIN Library l ON b.LibraryID = l.LibraryID
LEFT JOIN Payment p ON lo.LoanID = p.LoanID
WHERE lo.Status = 'Overdue'
GROUP BY m.Full_Name, m.Phone_Number, b.Title, l.Name, lo.Due_Date;
```

100 %

Results | Messages

	Member_Name	Phone_Number	Book_Title	Library_Name	Days_Overdue	Fine_Paid
1	Sara Al Maskari	9711001122	Data Structures in Python	Central Library	781	0.00

4. Staff Performance Overview



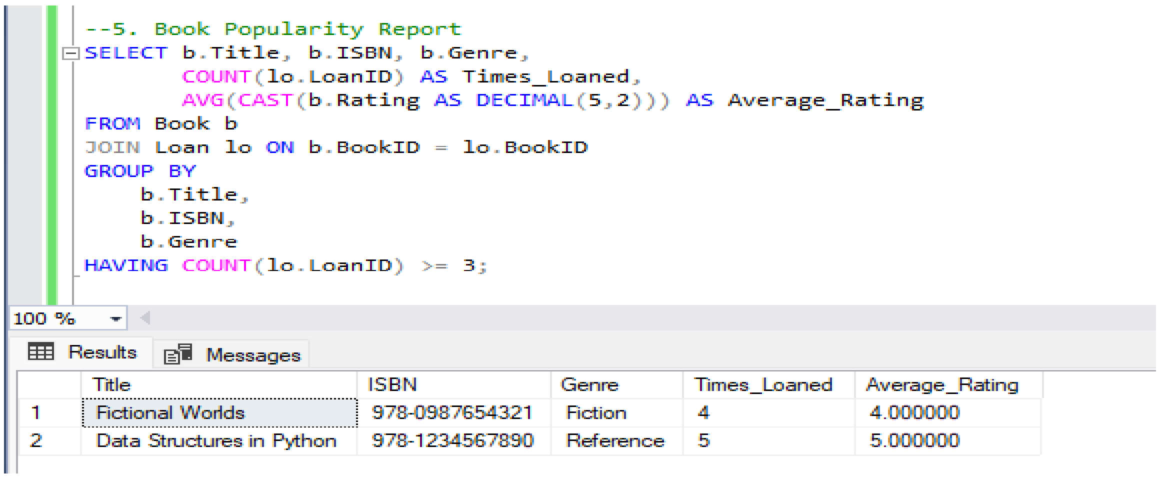
```sql
--4. Staff Performance Overview
SELECT l.Name AS Library_Name, s.Full_Name AS Staff_Name, s.Position,
       COUNT(b.BookID) AS Books_Managed
FROM Staff s
JOIN Library l ON s.LibraryID = l.LibraryID
LEFT JOIN Book b ON l.LibraryID = b.LibraryID
GROUP BY l.Name, s.Full_Name, s.Position;
```

100 %

Results | Messages

	Library_Name	Staff_Name	Position	Books_Managed
1	Al Hajar Library	Fatima Al Harthy	Assistant Librarian	21
2	Central Library	Ahmed Al Balushi	Librarian	24
3	Sultan Qaboos Library	Khalid Al Siyabi	Library Manager	21

5. Book Popularity Report



```sql
--5. Book Popularity Report
SELECT b.Title, b.ISBN, b.Genre,
       COUNT(lo.LoanID) AS Times_Loaned,
       AVG(CAST(b.Rating AS DECIMAL(5,2))) AS Average_Rating
FROM Book b
JOIN Loan lo ON b.BookID = lo.BookID
GROUP BY
    b.Title,
    b.ISBN,
    b.Genre
HAVING COUNT(lo.LoanID) >= 3;
```

100 %

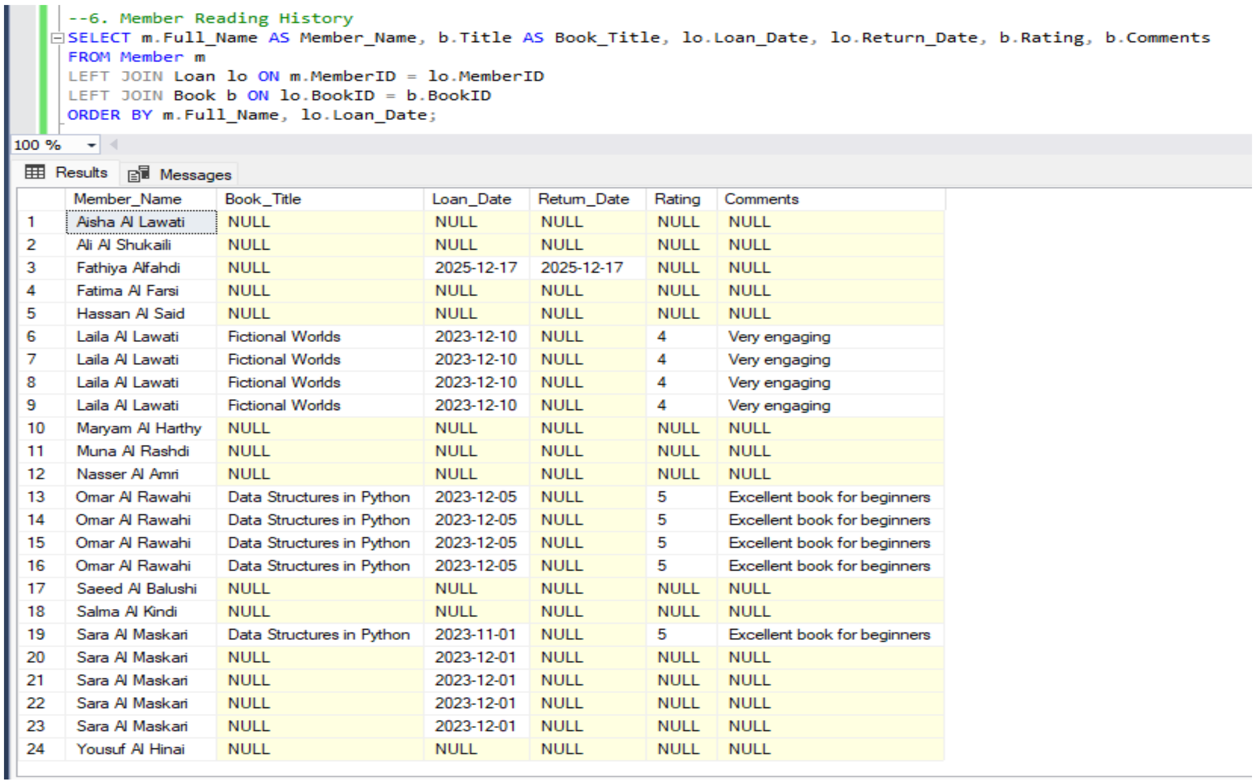
Results | Messages

	Title	ISBN	Genre	Times_Loaned	Average_Rating
1	Fictional Worlds	978-0987654321	Fiction	4	4.000000
2	Data Structures in Python	978-1234567890	Reference	5	5.000000

Explanation:

Joined Book and Loan tables and used COUNT() with GROUP BY to calculate how many times each book was loaned. The HAVING clause filters books loaned at least three times, while AVG() computes the average review rating.

6. Member Reading History



```sql
--6. Member Reading History
SELECT m.Full_Name AS Member_Name, b.Title AS Book_Title, lo.Loan_Date, lo.Return_Date, b.Rating, b.Comments
FROM Member m
LEFT JOIN Loan lo ON m.MemberID = lo.MemberID
LEFT JOIN Book b ON lo.BookID = b.BookID
ORDER BY m.Full_Name, lo.Loan_Date;
```

100 %

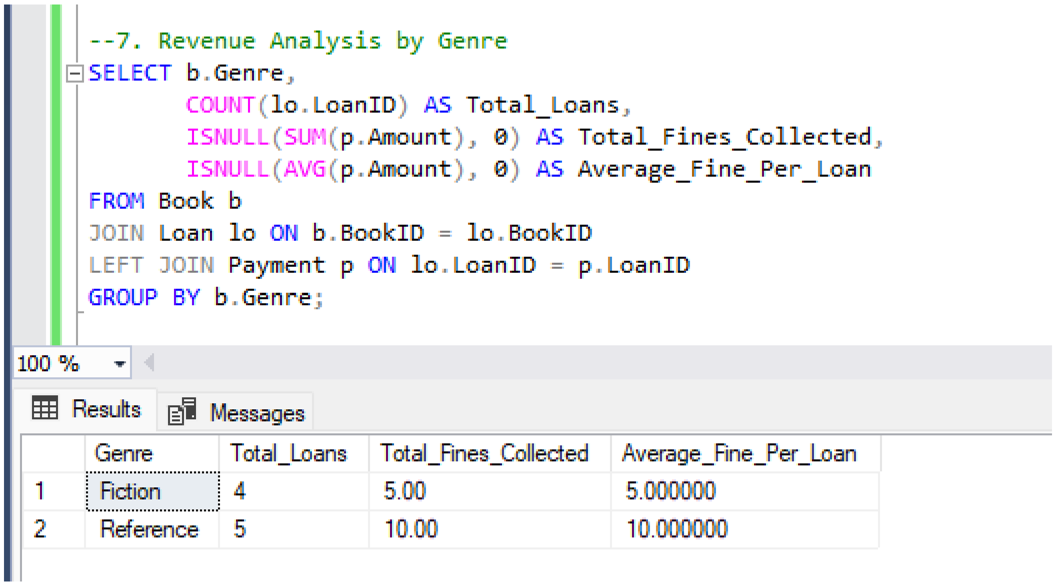
Results | Messages

	Member_Name	Book_Title	Loan_Date	Return_Date	Rating	Comments
1	Aisha Al Lawati	NULL	NULL	NULL	NULL	NULL
2	Ali Al Shukaili	NULL	NULL	NULL	NULL	NULL
3	Fathiya Alfahdi	NULL	2025-12-17	2025-12-17	NULL	NULL
4	Fatima Al Farsi	NULL	NULL	NULL	NULL	NULL
5	Hassan Al Said	NULL	NULL	NULL	NULL	NULL
6	Laila Al Lawati	Fictional Worlds	2023-12-10	NULL	4	Very engaging
7	Laila Al Lawati	Fictional Worlds	2023-12-10	NULL	4	Very engaging
8	Laila Al Lawati	Fictional Worlds	2023-12-10	NULL	4	Very engaging
9	Laila Al Lawati	Fictional Worlds	2023-12-10	NULL	4	Very engaging
10	Maryam Al Harthy	NULL	NULL	NULL	NULL	NULL
11	Muna Al Rashdi	NULL	NULL	NULL	NULL	NULL
12	Nasser Al Amri	NULL	NULL	NULL	NULL	NULL
13	Omar Al Rawahi	Data Structures in Python	2023-12-05	NULL	5	Excellent book for beginners
14	Omar Al Rawahi	Data Structures in Python	2023-12-05	NULL	5	Excellent book for beginners
15	Omar Al Rawahi	Data Structures in Python	2023-12-05	NULL	5	Excellent book for beginners
16	Omar Al Rawahi	Data Structures in Python	2023-12-05	NULL	5	Excellent book for beginners
17	Saeed Al Balushi	NULL	NULL	NULL	NULL	NULL
18	Salma Al Kindi	NULL	NULL	NULL	NULL	NULL
19	Sara Al Maskari	Data Structures in Python	2023-11-01	NULL	5	Excellent book for beginners
20	Sara Al Maskari	NULL	2023-12-01	NULL	NULL	NULL
21	Sara Al Maskari	NULL	2023-12-01	NULL	NULL	NULL
22	Sara Al Maskari	NULL	2023-12-01	NULL	NULL	NULL
23	Sara Al Maskari	NULL	2023-12-01	NULL	NULL	NULL
24	Yousuf Al Hinai	NULL	NULL	NULL	NULL	NULL

Explanation:

The result may contain repeated member names or book titles because each row represents a separate loan transaction. This correctly shows the complete borrowing history, including multiple books or repeated borrowings by the same member.

7. Revenue Analysis by Genre



```sql
--7. Revenue Analysis by Genre
SELECT b.Genre,
       COUNT(lo.LoanID) AS Total_Loans,
       ISNULL(SUM(p.Amount), 0) AS Total_Fines_Collected,
       ISNULL(AVG(p.Amount), 0) AS Average_Fine_Per_Loan
FROM Book b
JOIN Loan lo ON b.BookID = lo.BookID
LEFT JOIN Payment p ON lo.LoanID = p.LoanID
GROUP BY b.Genre;
```

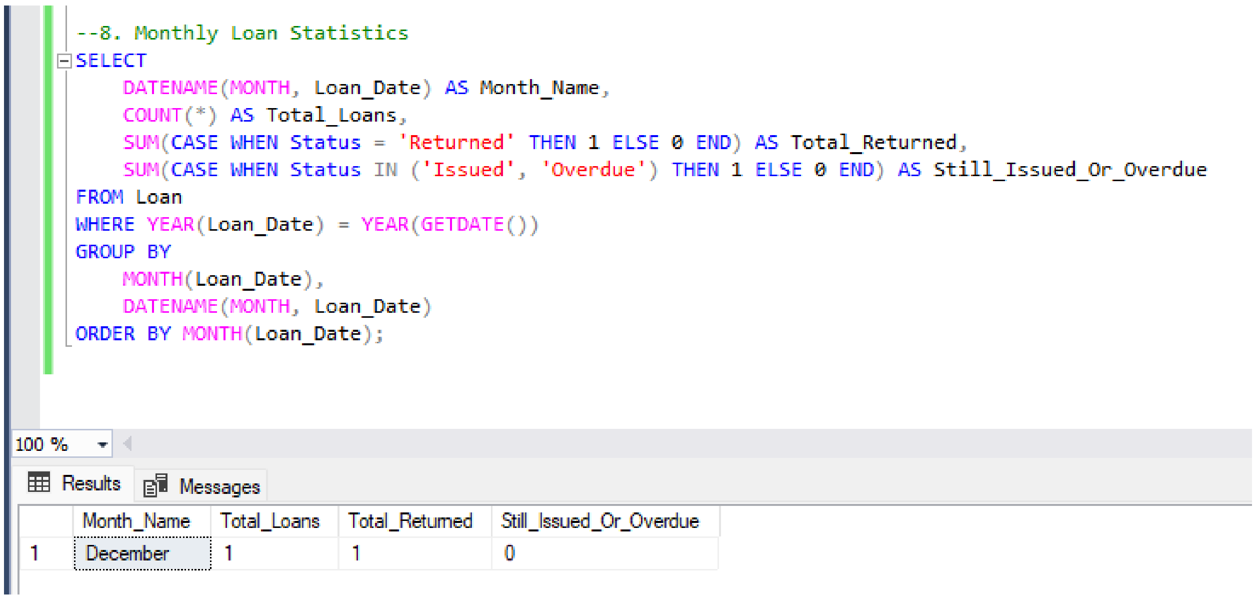
	Genre	Total_Loans	Total_Fines_Collected	Average_Fine_Per_Loan
1	Fiction	4	5.00	5.000000
2	Reference	5	10.00	10.000000

Explanation:

Joined Book, Loan, and Payment tables and grouped results by genre. Aggregate functions (SUM, COUNT, AVG) were applied to calculate total fines collected, number of loans, and average fine per loan for each genre.

Section 2: Aggregate Functions and Grouping

8. Monthly Loan Statistics



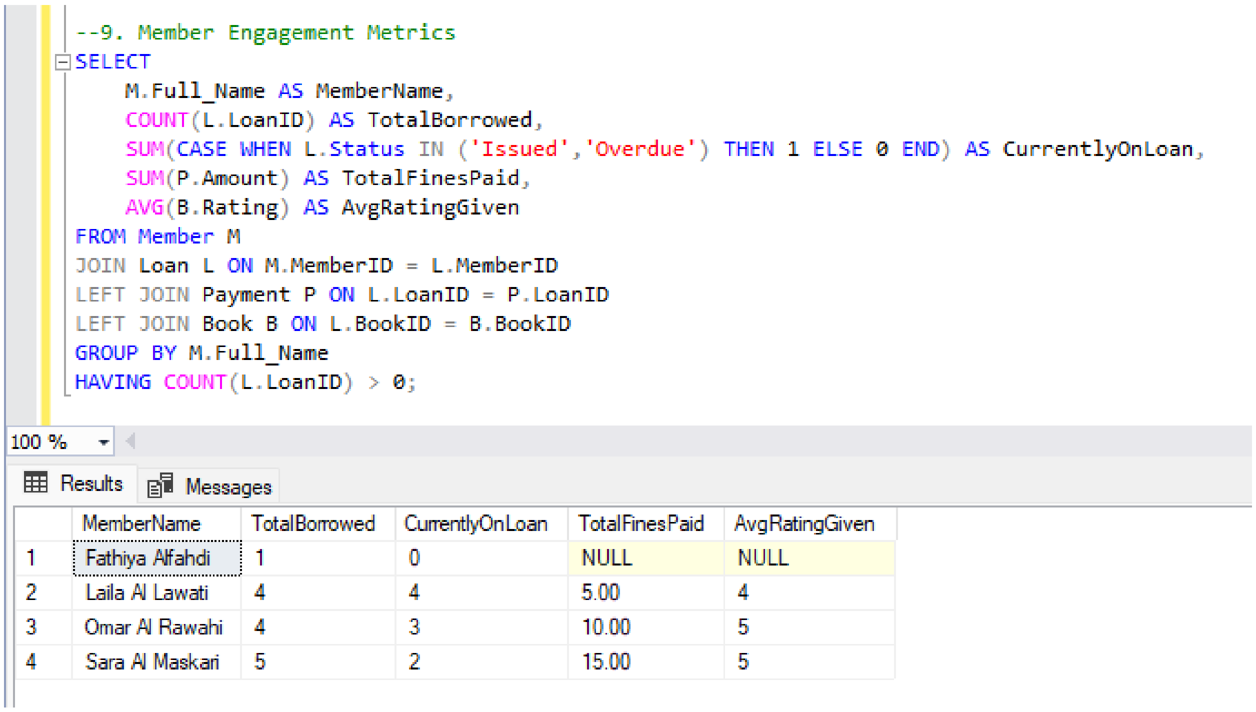
```
--8. Monthly Loan Statistics
SELECT
    DATENAME(MONTH, Loan_Date) AS Month_Name,
    COUNT(*) AS Total_Loans,
    SUM(CASE WHEN Status = 'Returned' THEN 1 ELSE 0 END) AS Total_Returned,
    SUM(CASE WHEN Status IN ('Issued', 'Overdue') THEN 1 ELSE 0 END) AS Still_Issued_Or_Overdue
FROM Loan
WHERE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

100 %

Results | Messages

	Month_Name	Total_Loans	Total_Returned	Still_Issued_Or_Overdue
1	December	1	1	0

9. Member Engagement Metrics



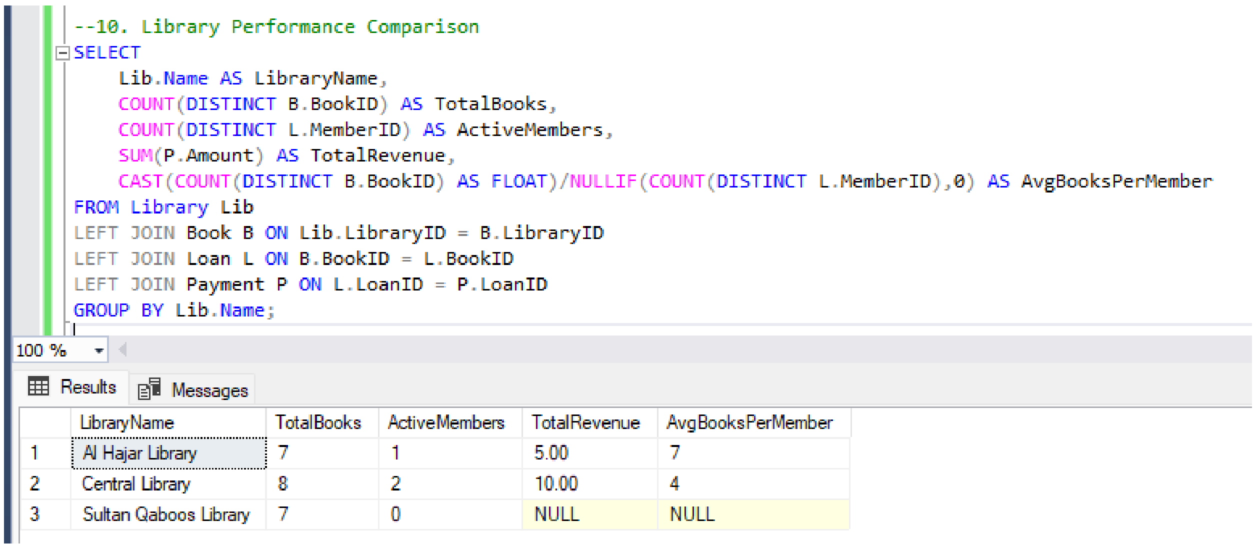
```
--9. Member Engagement Metrics
SELECT
    M.Full_Name AS MemberName,
    COUNT(L.LoanID) AS TotalBorrowed,
    SUM(CASE WHEN L.Status IN ('Issued','Overdue') THEN 1 ELSE 0 END) AS CurrentlyOnLoan,
    SUM(P.Amount) AS TotalFinesPaid,
    AVG(B.Rating) AS AvgRatingGiven
FROM Member M
JOIN Loan L ON M.MemberID = L.MemberID
LEFT JOIN Payment P ON L.LoanID = P.LoanID
LEFT JOIN Book B ON L.BookID = B.BookID
GROUP BY M.Full_Name
HAVING COUNT(L.LoanID) > 0;
```

100 %

Results | Messages

	MemberName	TotalBorrowed	CurrentlyOnLoan	TotalFinesPaid	AvgRatingGiven
1	Fathiya Alfahdi	1	0	NULL	NULL
2	Laila Al Lawati	4	4	5.00	4
3	Omar Al Rawahi	4	3	10.00	5
4	Sara Al Maskari	5	2	15.00	5

10. Library Performance Comparison



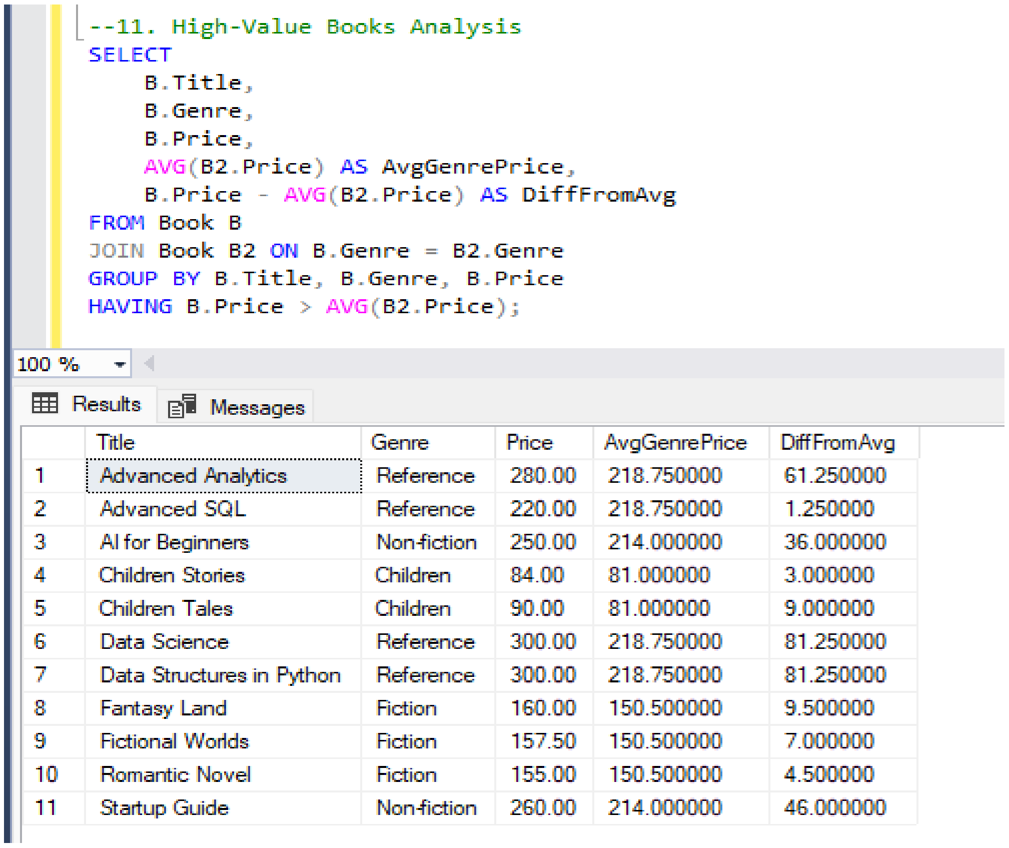
```sql
--10. Library Performance Comparison
SELECT
    Lib.Name AS LibraryName,
    COUNT(DISTINCT B.BookID) AS TotalBooks,
    COUNT(DISTINCT L.MemberID) AS ActiveMembers,
    SUM(P.Amount) AS TotalRevenue,
    CAST(COUNT(DISTINCT B.BookID) AS FLOAT)/NULLIF(COUNT(DISTINCT L.MemberID),0) AS AvgBooksPerMember
FROM Library Lib
LEFT JOIN Book B ON Lib.LibraryID = B.LibraryID
LEFT JOIN Loan L ON B.BookID = L.BookID
LEFT JOIN Payment P ON L.LoanID = P.LoanID
GROUP BY Lib.Name;
```

100 %

Results | Messages

	LibraryName	TotalBooks	ActiveMembers	TotalRevenue	AvgBooksPerMember
1	Al Hajar Library	7	1	5.00	7
2	Central Library	8	2	10.00	4
3	Sultan Qaboos Library	7	0	NULL	NULL

11. High-Value Books Analysis



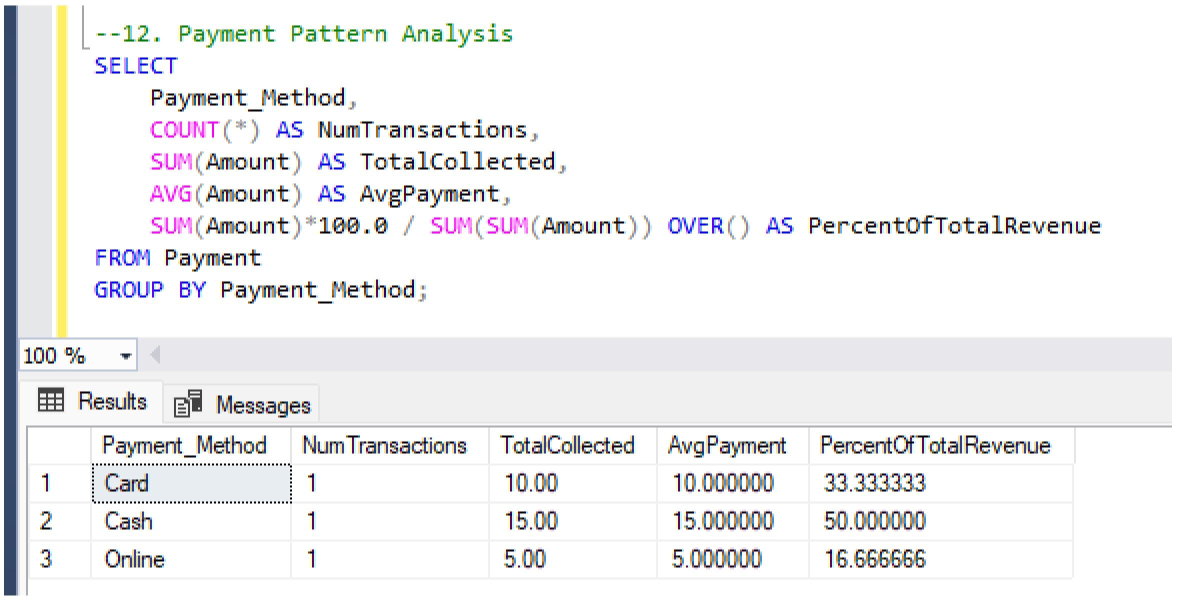
```sql
--11. High-Value Books Analysis
SELECT
    B.Title,
    B.Genre,
    B.Price,
    AVG(B2.Price) AS AvgGenrePrice,
    B.Price - AVG(B2.Price) AS DiffFromAvg
FROM Book B
JOIN Book B2 ON B.Genre = B2.Genre
GROUP BY B.Title, B.Genre, B.Price
HAVING B.Price > AVG(B2.Price);
```

100 %

Results | Messages

	Title	Genre	Price	AvgGenrePrice	DiffFromAvg
1	Advanced Analytics	Reference	280.00	218.750000	61.250000
2	Advanced SQL	Reference	220.00	218.750000	1.250000
3	AI for Beginners	Non-fiction	250.00	214.000000	36.000000
4	Children Stories	Children	84.00	81.000000	3.000000
5	Children Tales	Children	90.00	81.000000	9.000000
6	Data Science	Reference	300.00	218.750000	81.250000
7	Data Structures in Python	Reference	300.00	218.750000	81.250000
8	Fantasy Land	Fiction	160.00	150.500000	9.500000
9	Fictional Worlds	Fiction	157.50	150.500000	7.000000
10	Romantic Novel	Fiction	155.00	150.500000	4.500000
11	Startup Guide	Non-fiction	260.00	214.000000	46.000000

12. Payment Pattern Analysis

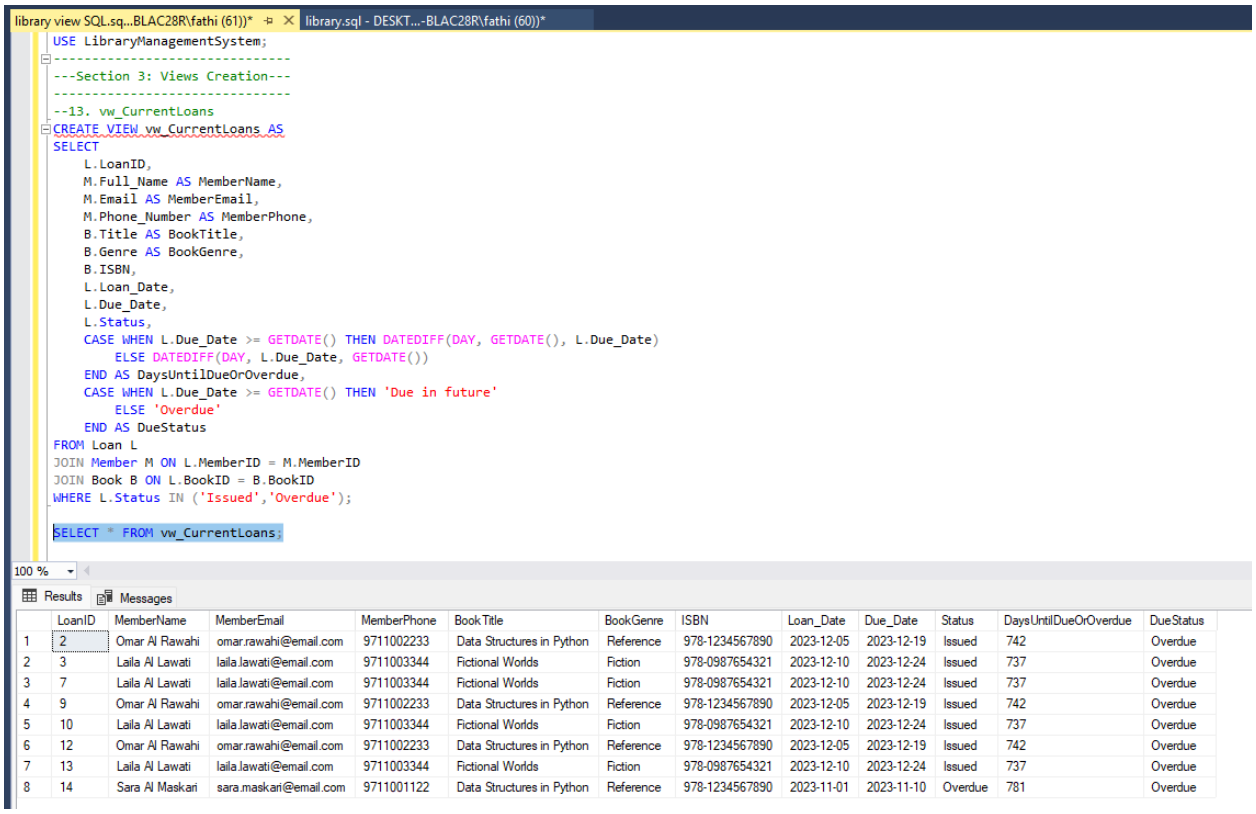


```
--12. Payment Pattern Analysis
SELECT
    Payment_Method,
    COUNT(*) AS NumTransactions,
    SUM(Amount) AS TotalCollected,
    AVG(Amount) AS AvgPayment,
    SUM(Amount)*100.0 / SUM(SUM(Amount)) OVER() AS PercentOfTotalRevenue
FROM Payment
GROUP BY Payment_Method;
```

	Payment_Method	NumTransactions	TotalCollected	AvgPayment	PercentOfTotalRevenue
1	Card	1	10.00	10.000000	33.333333
2	Cash	1	15.00	15.000000	50.000000
3	Online	1	5.00	5.000000	16.666666

Section 3: Views Creation

13. vw_CurrentLoans



library view SQL.sq...BLAC28R\fathi (61))* | library.sql - DESKT...-BLAC28R\fathi (60))*

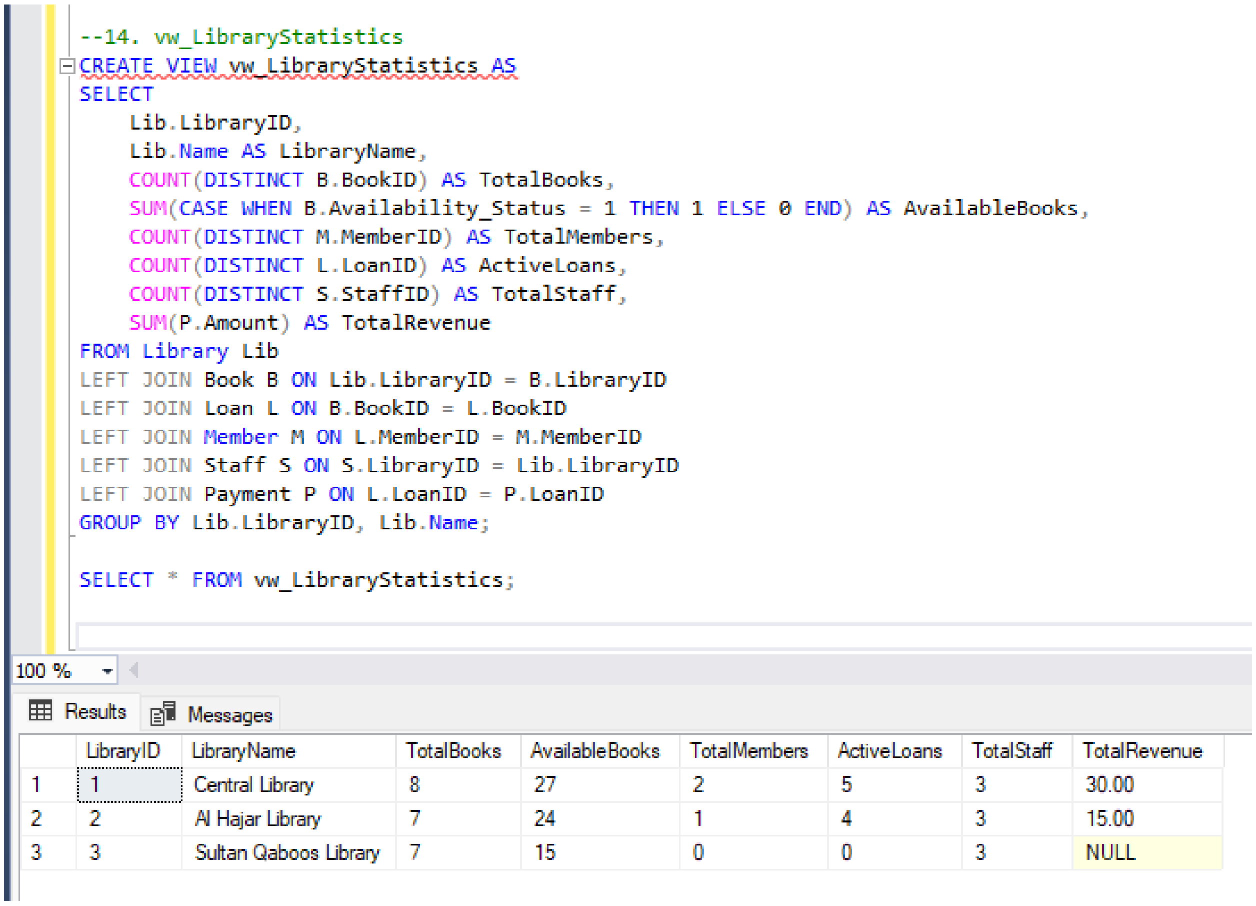
```sql
USE LibraryManagementSystem;
------------------------------
---Section 3: Views Creation---
------------------------------
--13. vw_CurrentLoans
CREATE VIEW vw_CurrentLoans AS
SELECT
    L.LoanID,
    M.Full_Name AS MemberName,
    M.Email AS MemberEmail,
    M.Phone_Number AS MemberPhone,
    B.Title AS BookTitle,
    B.Genre AS BookGenre,
    B.ISBN,
    L.Loan_Date,
    L.Due_Date,
    L.Status,
    CASE WHEN L.Due_Date >= GETDATE() THEN DATEDIFF(DAY, GETDATE(), L.Due_Date)
        ELSE DATEDIFF(DAY, L.Due_Date, GETDATE())
    END AS DaysUntilDueOrOverdue,
    CASE WHEN L.Due_Date >= GETDATE() THEN 'Due in future'
        ELSE 'Overdue'
    END AS DueStatus
FROM Loan L
JOIN Member M ON L.MemberID = M.MemberID
JOIN Book B ON L.BookID = B.BookID
WHERE L.Status IN ('Issued','Overdue');

SELECT * FROM vw_CurrentLoans;
```

100 %

Results | Messages

	LoanID	MemberName	MemberEmail	MemberPhone	BookTitle	BookGenre	ISBN	Loan_Date	Due_Date	Status	DaysUntilDueOrOverdue	DueStatus
1	2	Omar Al Rawahi	omar.rawahi@email.com	9711002233	Data Structures in Python	Reference	978-1234567890	2023-12-05	2023-12-19	Issued	742	Overdue
2	3	Laila Al Lawati	laila.lawati@email.com	9711003344	Fictional Worlds	Fiction	978-0987654321	2023-12-10	2023-12-24	Issued	737	Overdue
3	7	Laila Al Lawati	laila.lawati@email.com	9711003344	Fictional Worlds	Fiction	978-0987654321	2023-12-10	2023-12-24	Issued	737	Overdue
4	9	Omar Al Rawahi	omar.rawahi@email.com	9711002233	Data Structures in Python	Reference	978-1234567890	2023-12-05	2023-12-19	Issued	742	Overdue
5	10	Laila Al Lawati	laila.lawati@email.com	9711003344	Fictional Worlds	Fiction	978-0987654321	2023-12-10	2023-12-24	Issued	737	Overdue
6	12	Omar Al Rawahi	omar.rawahi@email.com	9711002233	Data Structures in Python	Reference	978-1234567890	2023-12-05	2023-12-19	Issued	742	Overdue
7	13	Laila Al Lawati	laila.lawati@email.com	9711003344	Fictional Worlds	Fiction	978-0987654321	2023-12-10	2023-12-24	Issued	737	Overdue
8	14	Sara Al Maskari	sara.maskari@email.com	9711001122	Data Structures in Python	Reference	978-1234567890	2023-11-01	2023-11-10	Overdue	781	Overdue

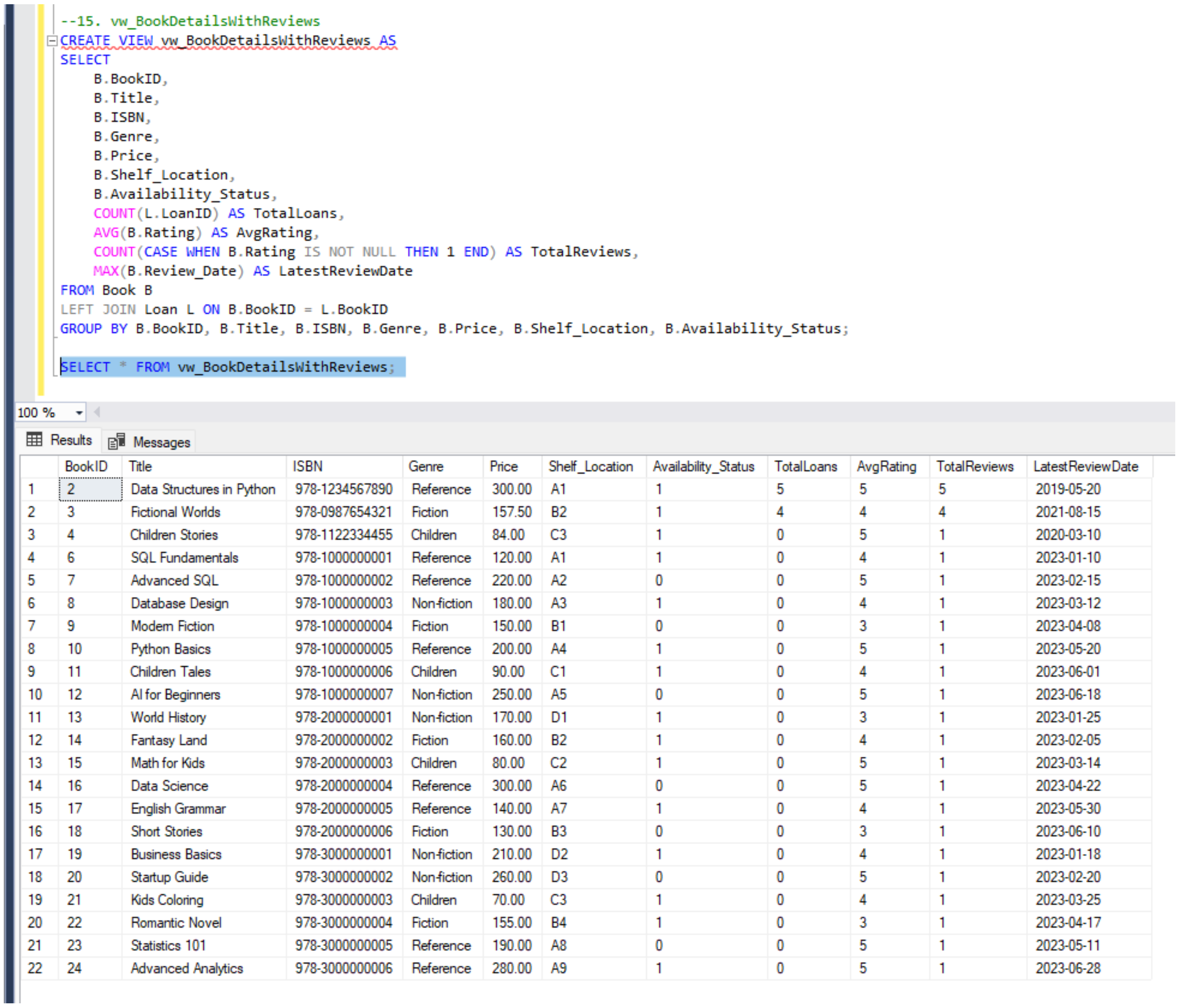
14. vw_LibraryStatistics



```
--14. vw_LibraryStatistics
CREATE VIEW vw_LibraryStatistics AS
SELECT
    Lib.LibraryID,
    Lib.Name AS LibraryName,
    COUNT(DISTINCT B.BookID) AS TotalBooks,
    SUM(CASE WHEN B.Availability_Status = 1 THEN 1 ELSE 0 END) AS AvailableBooks,
    COUNT(DISTINCT M.MemberID) AS TotalMembers,
    COUNT(DISTINCT L.LoanID) AS ActiveLoans,
    COUNT(DISTINCT S.StaffID) AS TotalStaff,
    SUM(P.Amount) AS TotalRevenue
FROM Library Lib
LEFT JOIN Book B ON Lib.LibraryID = B.LibraryID
LEFT JOIN Loan L ON B.BookID = L.BookID
LEFT JOIN Member M ON L.MemberID = M.MemberID
LEFT JOIN Staff S ON S.LibraryID = Lib.LibraryID
LEFT JOIN Payment P ON L.LoanID = P.LoanID
GROUP BY Lib.LibraryID, Lib.Name;

SELECT * FROM vw_LibraryStatistics;
```

	LibraryID	LibraryName	TotalBooks	AvailableBooks	TotalMembers	ActiveLoans	TotalStaff	TotalRevenue
1	1	Central Library	8	27	2	5	3	30.00
2	2	Al Hajar Library	7	24	1	4	3	15.00
3	3	Sultan Qaboos Library	7	15	0	0	3	NULL

15. vw_BookDetailsWithReviews



```sql
--15. vw_BookDetailsWithReviews
CREATE VIEW vw_BookDetailsWithReviews AS
SELECT
    B.BookID,
    B.Title,
    B.ISBN,
    B.Genre,
    B.Price,
    B.Shelf_Location,
    B.Availability_Status,
    COUNT(L.LoanID) AS TotalLoans,
    AVG(B.Rating) AS AvgRating,
    COUNT(CASE WHEN B.Rating IS NOT NULL THEN 1 END) AS TotalReviews,
    MAX(B.Review_Date) AS LatestReviewDate
FROM Book B
LEFT JOIN Loan L ON B.BookID = L.BookID
GROUP BY B.BookID, B.Title, B.ISBN, B.Genre, B.Price, B.Shelf_Location, B.Availability_Status;

SELECT * FROM vw_BookDetailsWithReviews;
```

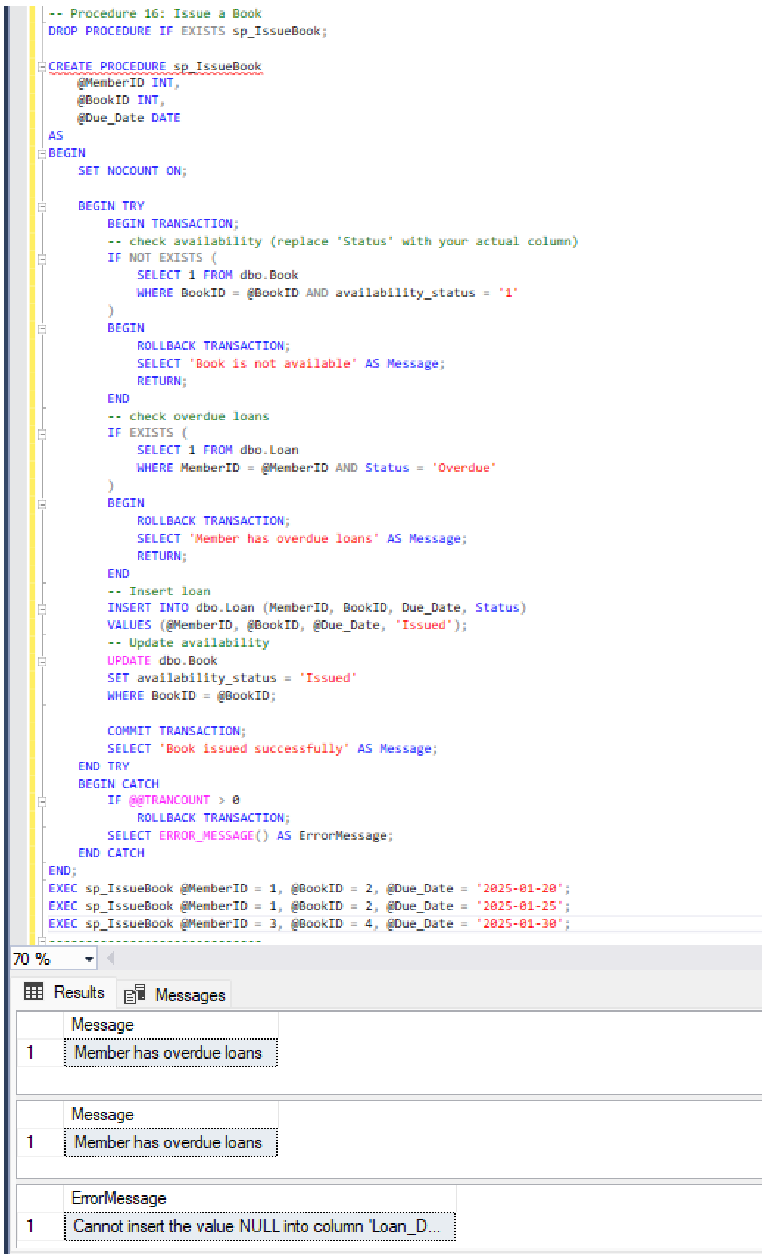
100 %

Results | Messages

	BookID	Title	ISBN	Genre	Price	Shelf_Location	Availability_Status	TotalLoans	AvgRating	TotalReviews	LatestReviewDate
1	2	Data Structures in Python	978-1234567890	Reference	300.00	A1	1	5	5	5	2019-05-20
2	3	Fictional Worlds	978-0987654321	Fiction	157.50	B2	1	4	4	4	2021-08-15
3	4	Children Stories	978-1122334455	Children	84.00	C3	1	0	5	1	2020-03-10
4	6	SQL Fundamentals	978-1000000001	Reference	120.00	A1	1	0	4	1	2023-01-10
5	7	Advanced SQL	978-1000000002	Reference	220.00	A2	0	0	5	1	2023-02-15
6	8	Database Design	978-1000000003	Non-fiction	180.00	A3	1	0	4	1	2023-03-12
7	9	Modern Fiction	978-1000000004	Fiction	150.00	B1	0	0	3	1	2023-04-08
8	10	Python Basics	978-1000000005	Reference	200.00	A4	1	0	5	1	2023-05-20
9	11	Children Tales	978-1000000006	Children	90.00	C1	1	0	4	1	2023-06-01
10	12	AI for Beginners	978-1000000007	Non-fiction	250.00	A5	0	0	5	1	2023-06-18
11	13	World History	978-2000000001	Non-fiction	170.00	D1	1	0	3	1	2023-01-25
12	14	Fantasy Land	978-2000000002	Fiction	160.00	B2	1	0	4	1	2023-02-05
13	15	Math for Kids	978-2000000003	Children	80.00	C2	1	0	5	1	2023-03-14
14	16	Data Science	978-2000000004	Reference	300.00	A6	0	0	5	1	2023-04-22
15	17	English Grammar	978-2000000005	Reference	140.00	A7	1	0	4	1	2023-05-30
16	18	Short Stories	978-2000000006	Fiction	130.00	B3	0	0	3	1	2023-06-10
17	19	Business Basics	978-3000000001	Non-fiction	210.00	D2	1	0	4	1	2023-01-18
18	20	Startup Guide	978-3000000002	Non-fiction	260.00	D3	0	0	5	1	2023-02-20
19	21	Kids Coloring	978-3000000003	Children	70.00	C3	1	0	4	1	2023-03-25
20	22	Romantic Novel	978-3000000004	Fiction	155.00	B4	1	0	3	1	2023-04-17
21	23	Statistics 101	978-3000000005	Reference	190.00	A8	0	0	5	1	2023-05-11
22	24	Advanced Analytics	978-3000000006	Reference	280.00	A9	1	0	5	1	2023-06-28

Section 4: Stored Procedures

16. sp_IssueBook



```sql
-- Procedure 16: Issue a Book
DROP PROCEDURE IF EXISTS sp_IssueBook;

CREATE PROCEDURE sp_IssueBook
    @MemberID INT,
    @BookID INT,
    @Due_Date DATE
AS
BEGIN
    SET NOCOUNT ON;

    BEGIN TRY
        BEGIN TRANSACTION;
        -- check availability (replace 'Status' with your actual column)
        IF NOT EXISTS (
            SELECT 1 FROM dbo.Book
            WHERE BookID = @BookID AND availability_status = '1'
        )
        BEGIN
            ROLLBACK TRANSACTION;
            SELECT 'Book is not available' AS Message;
            RETURN;
        END
        -- check overdue loans
        IF EXISTS (
            SELECT 1 FROM dbo.Loan
            WHERE MemberID = @MemberID AND Status = 'Overdue'
        )
        BEGIN
            ROLLBACK TRANSACTION;
            SELECT 'Member has overdue loans' AS Message;
            RETURN;
        END
        -- Insert loan
        INSERT INTO dbo.Loan (MemberID, BookID, Due_Date, Status)
        VALUES (@MemberID, @BookID, @Due_Date, 'Issued');
        -- Update availability
        UPDATE dbo.Book
        SET availability_status = 'Issued'
        WHERE BookID = @BookID;

        COMMIT TRANSACTION;
        SELECT 'Book issued successfully' AS Message;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;
        SELECT ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
END;
EXEC sp_IssueBook @MemberID = 1, @BookID = 2, @Due_Date = '2025-01-20';
EXEC sp_IssueBook @MemberID = 1, @BookID = 2, @Due_Date = '2025-01-25';
EXEC sp_IssueBook @MemberID = 3, @BookID = 4, @Due_Date = '2025-01-30';
-----------------------------
```

70 %

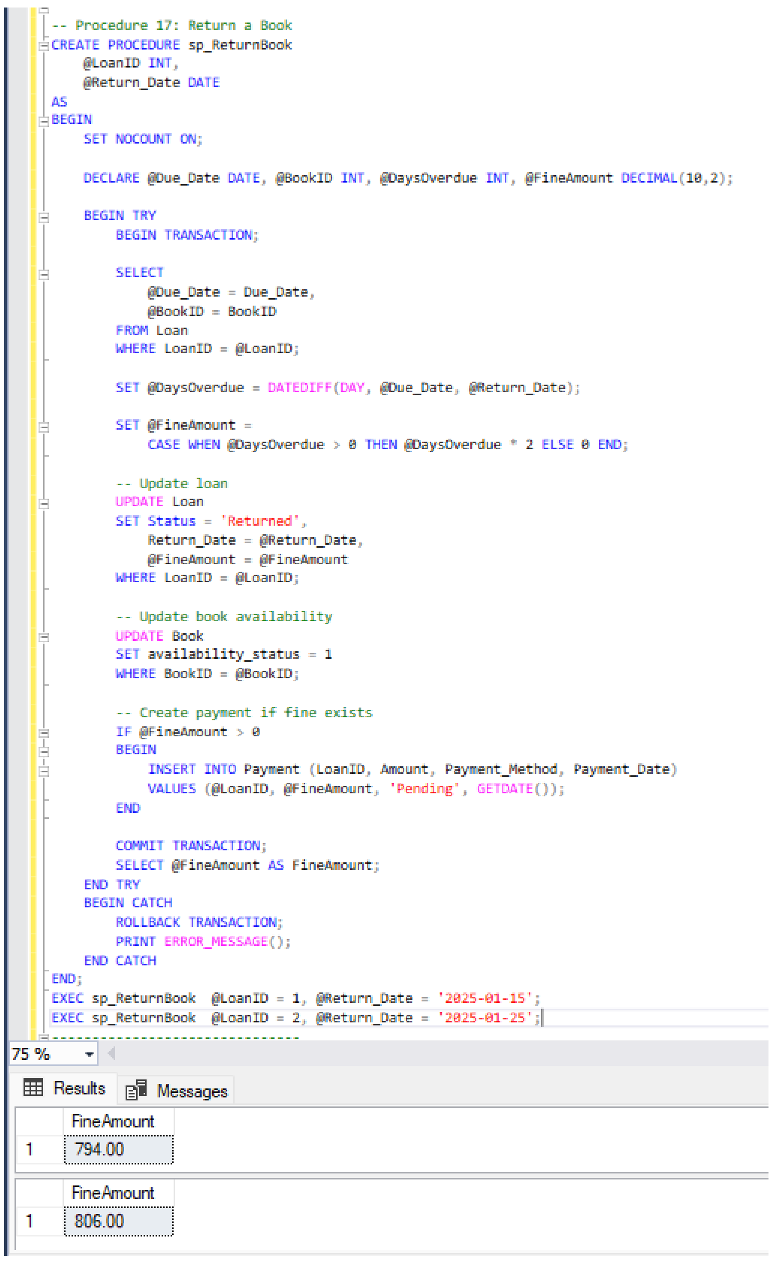
Results | Messages

	Message
1	Member has overdue loans

	Message
1	Member has overdue loans

	ErrorMessage
1	Cannot insert the value NULL into column 'Loan_D...

17. sp_ReturnBook



```sql
-- Procedure 17: Return a Book
CREATE PROCEDURE sp_ReturnBook
    @LoanID INT,
    @Return_Date DATE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @Due_Date DATE, @BookID INT, @DaysOverdue INT, @FineAmount DECIMAL(10,2);

    BEGIN TRY
        BEGIN TRANSACTION;

        SELECT
            @Due_Date = Due_Date,
            @BookID = BookID
        FROM Loan
        WHERE LoanID = @LoanID;

        SET @DaysOverdue = DATEDIFF(DAY, @Due_Date, @Return_Date);

        SET @FineAmount =
            CASE WHEN @DaysOverdue > 0 THEN @DaysOverdue * 2 ELSE 0 END;

        -- Update loan
        UPDATE Loan
        SET Status = 'Returned',
            Return_Date = @Return_Date,
            @FineAmount = @FineAmount
        WHERE LoanID = @LoanID;

        -- Update book availability
        UPDATE Book
        SET availability_status = 1
        WHERE BookID = @BookID;

        -- Create payment if fine exists
        IF @FineAmount > 0
        BEGIN
            INSERT INTO Payment (LoanID, Amount, Payment_Method, Payment_Date)
            VALUES (@LoanID, @FineAmount, 'Pending', GETDATE());
        END

        COMMIT TRANSACTION;
        SELECT @FineAmount AS FineAmount;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        PRINT ERROR_MESSAGE();
    END CATCH
END;
EXEC sp_ReturnBook  @LoanID = 1, @Return_Date = '2025-01-15';
EXEC sp_ReturnBook  @LoanID = 2, @Return_Date = '2025-01-25';
```

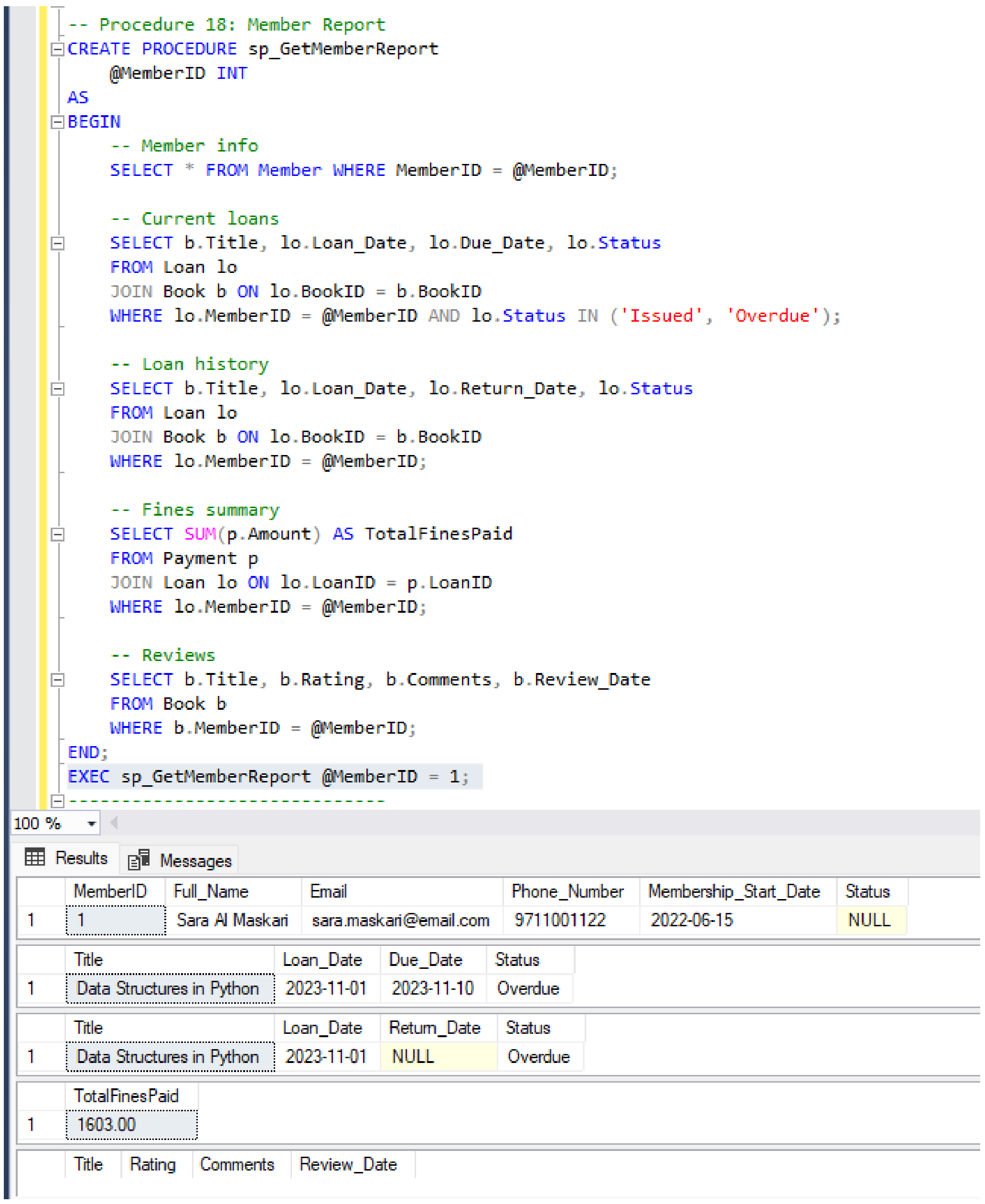
75 %

Results | Messages

	FineAmount
1	794.00

	FineAmount
1	806.00

18. sp_GetMemberReport



```
-- Procedure 18: Member Report
CREATE PROCEDURE sp_GetMemberReport
    @MemberID INT
AS
BEGIN
    -- Member info
    SELECT * FROM Member WHERE MemberID = @MemberID;

    -- Current loans
    SELECT b.Title, lo.Loan_Date, lo.Due_Date, lo.Status
    FROM Loan lo
    JOIN Book b ON lo.BookID = b.BookID
    WHERE lo.MemberID = @MemberID AND lo.Status IN ('Issued', 'Overdue');

    -- Loan history
    SELECT b.Title, lo.Loan_Date, lo.Return_Date, lo.Status
    FROM Loan lo
    JOIN Book b ON lo.BookID = b.BookID
    WHERE lo.MemberID = @MemberID;

    -- Fines summary
    SELECT SUM(p.Amount) AS TotalFinesPaid
    FROM Payment p
    JOIN Loan lo ON lo.LoanID = p.LoanID
    WHERE lo.MemberID = @MemberID;

    -- Reviews
    SELECT b.Title, b.Rating, b.Comments, b.Review_Date
    FROM Book b
    WHERE b.MemberID = @MemberID;
END;
EXEC sp_GetMemberReport @MemberID = 1;
------------------------------
```

100 %

Results | Messages

	MemberID	Full_Name	Email	Phone_Number	Membership_Start_Date	Status
1	1	Sara Al Maskari	sara.maskari@email.com	9711001122	2022-06-15	NULL

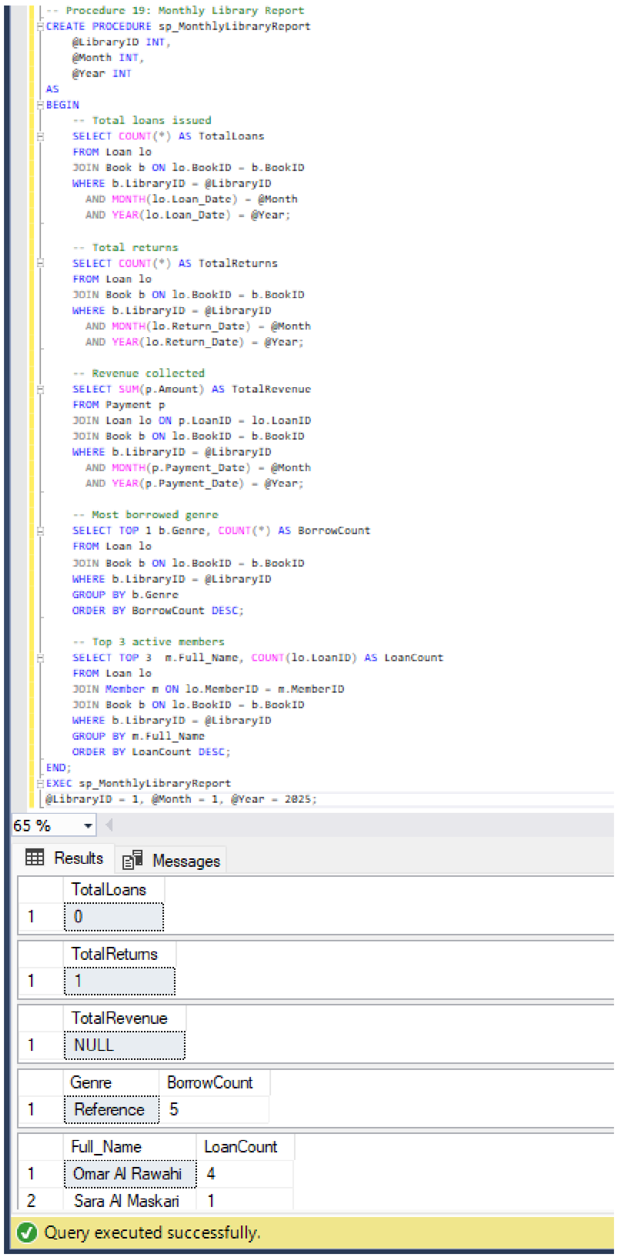
	Title	Loan_Date	Due_Date	Status
1	Data Structures in Python	2023-11-01	2023-11-10	Overdue

	Title	Loan_Date	Return_Date	Status
1	Data Structures in Python	2023-11-01	NULL	Overdue

	TotalFinesPaid
1	1603.00

	Title	Rating	Comments	Review_Date

19. sp_MonthlyLibraryReport



```sql
-- Procedure 19: Monthly Library Report
CREATE PROCEDURE sp_MonthlyLibraryReport
    @LibraryID INT,
    @Month INT,
    @Year INT
AS
BEGIN
    -- Total loans issued
    SELECT COUNT(*) AS TotalLoans
    FROM Loan lo
    JOIN Book b ON lo.BookID = b.BookID
    WHERE b.LibraryID = @LibraryID
      AND MONTH(lo.Loan_Date) = @Month
      AND YEAR(lo.Loan_Date) = @Year;

    -- Total returns
    SELECT COUNT(*) AS TotalReturns
    FROM Loan lo
    JOIN Book b ON lo.BookID = b.BookID
    WHERE b.LibraryID = @LibraryID
      AND MONTH(lo.Return_Date) = @Month
      AND YEAR(lo.Return_Date) = @Year;

    -- Revenue collected
    SELECT SUM(p.Amount) AS TotalRevenue
    FROM Payment p
    JOIN Loan lo ON p.LoanID = lo.LoanID
    JOIN Book b ON lo.BookID = b.BookID
    WHERE b.LibraryID = @LibraryID
      AND MONTH(p.Payment_Date) = @Month
      AND YEAR(p.Payment_Date) = @Year;

    -- Most borrowed genre
    SELECT TOP 1 b.Genre, COUNT(*) AS BorrowCount
    FROM Loan lo
    JOIN Book b ON lo.BookID = b.BookID
    WHERE b.LibraryID = @LibraryID
    GROUP BY b.Genre
    ORDER BY BorrowCount DESC;

    -- Top 3 active members
    SELECT TOP 3  m.Full_Name, COUNT(lo.LoanID) AS LoanCount
    FROM Loan lo
    JOIN Member m ON lo.MemberID = m.MemberID
    JOIN Book b ON lo.BookID = b.BookID
    WHERE b.LibraryID = @LibraryID
    GROUP BY m.Full_Name
    ORDER BY LoanCount DESC;
END;
EXEC sp_MonthlyLibraryReport
@LibraryID = 1, @Month = 1, @Year = 2025;
```

65 %

Results Messages

	TotalLoans
1	0

	TotalReturns
1	1

	TotalRevenue
1	NULL

	Genre	BorrowCount
1	Reference	5

	Full_Name	LoanCount
1	Omar Al Rawahi	4
2	Sara Al Maskari	1

Query executed successfully.