EECS 447 Project Part 6

Team SQLibrary

May 4, 2025

1 Given Query Execution and Validation

In this final phase of the project, we focus on executing and validating SQL queries that demonstrate meaningful insights into the Library Management System's data. Each subsection includes a description of the query, the SQL command issued, the expected output, actual execution result (with screenshot), and an explanation verifying correctness. There are 19 queries, each with an execution to showcase the library's capability. The first seven were provided, the subsequent twelve are our own. Our demonstration is scheduled for Wednesday, May 7, at 2 PM. These queries will be preconfigured to run.

Query 1: Fine Calculation

Description: Calculate the total fines owed by each member, summing only unpaid fines. The fine rate is established by a member's membertype, and is established in the tables.

Expected Output: A list of MemberIDs and the total amount they owe in unpaid fines.

SQL Command:

```
SELECT MemberID, SUM(Amount) AS TotalFine
FROM Fine
WHERE PaymentStatus = 'Unpaid'
GROUP BY MemberID;
```

Execution Result:

	MemberID	TotalFine
1	2304291e-5d72-4e1e-bb81-8d0fa417f773	8.11
2	2fcef34b-a11f-4f94-bba2-efdc7b582464	6.10
3	40cf5577-fd8d-4d9e-bef0-61e5bc601387	7.15
4	4e782040-ddc6-41a3-9ed1-bc2b54157763	6.58
5	800aff3c-7dc3-41fe-8569-9b679bef6c3b	5.96
6	9069871e-073c-4926-acfa-bb6a96a8b23f	5.04
7	97c15564-1a62-48f4-9d9a-f6475042696b	5.31
8	9b545725-d30e-492f-ac35-19b6f5117d7f	7.53
9	a59cb134-154d-42d0-85db-2c551387e40e	6.05
10	aa18715d-c041-4e2d-a528-bf0ea09f6c8b	6.20
11	ae13419f-ce14-45bb-b49c-cc144cc19f82	5.32
12	c00844d1-4ed6-4752-8c7a-c54dc09c745c	6.03
13	c39b445a-bbf4-46e3-bceb-f3bb9c02a5fa	7.15
14	e8c2144f-e6c0-4c00-b0a0-19d10fd0563b	5.54
15	eb35a2d8-196a-48a9-bd97-4dd7bb8da4c1	6.19
16	f7fd4372-186f-4d0a-9414-020901c67302	8.95
17	ff53f69f-1ed0-4f6b-ae69-5565dbf41a7f	5.92

Explanation: This query correctly sums the unpaid fines by member. The totals align with the raw data seen in the Fine table, and only unpaid rows are considered due to the WHERE clause. This can be checked with simple math, 24 Fines, 7 paid, 17 should be returned.

Query 2: Book Availability by Genre

Description: List all available books of a specific genre, e.g., 'Science Fiction'.

Expected Output: Books that are marked 'Available' and belong to the specified genre.

SQL Command:

```
SELECT B.Title, B.Author, B.Genre
FROM Book B
JOIN Item I ON B.ItemID = I.ItemID
WHERE I.Availability = 'Available' AND B.Genre = 'Science Fiction';
```

Execution Result:

	Title	Author	Genre
1	The Starborn Prophecy	Clara Granger	Science Fiction
2	The Violet Flame	Pietrek Simcock	Science Fiction
3	Echospire	Maeve Marston	Science Fiction

Explanation: Results show the three science fiction books that are availabile. The JOIN on Item is necessary to access availability status. We can confirm this is correct by referencing the Item table, which shows the available books being the only ones selected.

Query 3: Frequent Borrowers by Genre

Description: Find members who borrowed the most 'Mystery' books in the past 13 months.

Expected Output: A ranked list of members by the count of mystery genre book loans.

SQL Command:

```
SELECT L.MemberID, COUNT(*) AS BorrowCount
FROM Loan L
        JOIN Book B ON L.ItemID = B.ItemID
WHERE TRIM(B.Genre) = 'Mystery'
  AND L.LoanDate >= NOW() - INTERVAL 13 MONTH
GROUP BY L.MemberID
ORDER BY BorrowCount DESC;
```

Execution Result:

	MemberID	BorrowCount
1	abc998da-08a3-4c49-99ef-d9cde0fa35de	1
2	81cf6e5d-d7ca-4cae-9373-5665b664759d	1
3	968fa4ad-49f2-4ab5-b9d7-6ff23ad98dda	1
4	31816ee1-3995-4fdb-8e78-5b5339f31ef2	1

Explanation: The query correctly ranks members based on how many mystery books they borrowed over the past year, since our data was created with unique loans. We didn't want to overflow our data with single members making multiple loans, so each member having only one loan, and there being four of them, is correct.

Query 4: Books Due Within a Week

Description: Report books due in the next 7 days.

Expected Output: Loan records with DueDate between May 3, and May 9.

SQL Command:

```
SELECT I.ItemType, COALESCE(B.Title, D.Title, M.Title) AS Title, L.DueDate
FROM Loan L
        JOIN Item I ON L.ItemID = I.ItemID
        LEFT JOIN Book B ON L.ItemID = B.ItemID
        LEFT JOIN DigitalMedia D ON L.ItemID = D.ItemID
        LEFT JOIN Magazine M ON L.ItemID = M.ItemID
WHERE DATE(L.DueDate) BETWEEN '2024-05-03' AND '2024-05-09'
ORDER BY L.DueDate ASC;
```

Execution Result:

	ItemType	Title	DueDate
1	Digital Media	Beyond the Pixel	2024-05-03 12:00:00
2	Book	The Gilded Conspiracy	2024-05-03 12:00:00
3	Digital Media	Quantum Broadcast	2024-05-04 12:00:00
4	Book	Tides of the Nether	2024-05-04 12:00:00
5	Book	The Ember Codex	2024-05-04 12:00:00
6	Magazine	The Curator	2024-05-04 12:00:00
7	Magazine	The Lens	2024-05-04 12:00:00
8	Book	Tempest and Tranquility	2024-05-04 12:00:00
9	Digital Media	The Wired Revolution	2024-05-04 12:00:00
10	Book	The Obsidian Key	2024-05-04 12:00:00
11	Book	The Clockwork Dominion	2024-05-04 12:00:00
12	Book	Lanterns at Dusk	2024-05-04 12:00:00
13	Book	Ashes of the Oracle	2024-05-04 12:00:00
14	Magazine	Future Fabric	2024-05-04 12:00:00
15	Book	Cursed Horizons	2024-05-04 12:00:00
16	Digital Media	Megacities in Motion	2024-05-04 12:00:00
17	Book	Echospire	2024-05-04 12:00:00
18	Book	Echoes of the Lost Empire	2024-05-04 12:00:00
19	Magazine	Urban Sketch	2024-05-04 12:00:00
20	Book	The Starborn Prophecy	2024-05-04 12:00:00
21	Digital Media	Streaming Evolution	2024-05-05 12:00:00
22	Digital Media	Interlinked	2024-05-05 12:00:00
23	Book	Beneath Crimson Skies	2024-05-05 12:00:00
24	Digital Media	Cryptoverse Explained	2024-05-05 12:00:00
25	Book	Labyrinth of Echoes	2024-05-05 12:00:00
26	Magazine	Open Horizons	2024-05-05 12:00:00
27	Book	Grimoire of Night	2024-05-05 12:00:00
28	Book	Emberlight	2024-05-05 12:00:00
29	Book	Bloodlines of Arkenfall	2024-05-05 12:00:00
30	Book	The Last Archivist	2024-05-05 12:00:00

100 rows

Explanation: Loans due within the week of 5/3-5/9 are returned. Since our data was built around a dataset close to this week, it is all 100 records, as they fall within the selected range. So 100 rows is correct to see as the result.

Query 5: Members with Overdue Books

Description: Identify members with at least one overdue book, and list the titles.

Expected Output: Member names and corresponding overdue book titles.

SQL Command:

```
SELECT M.Name AS MemberName, B.Title AS BookTitle, L.DueDate
FROM Loan L
JOIN Member M ON L.MemberID = M.MemberID
```

```
JOIN Book B ON L.ItemID = B.ItemID
WHERE L.ReturnDate > L.DueDate OR (L.ReturnDate IS NULL AND L.DueDate < NOW())
ORDER BY M.Name;
```

Execution Result:

	MemberName	BookTitle	DueDate
1	Christie Zeplin	Phantom Tide	2024-05-06 12:00:00
2	Clarisse Water	Scrolls of the Ancients	2024-05-05 12:00:00
3	Dion Sawdy	The Last Archivist	2024-05-05 12:00:00
4	Flori Gibb	Echospire	2024-05-04 12:00:00
5	Gay Gatheral	Dagger of Truth	2024-05-05 12:00:00
6	Lauri Blezard	Path of the Hollowed	2024-05-05 12:00:00
7	Mellisent Litster	Thorns and Feathers	2024-05-06 12:00:00
8	Milton Buss	Emberlight	2024-05-05 12:00:00
9	Noella Stronough	Tales of the Glass Realm	2024-05-06 12:00:00
10	Sal Heaysman	The Violet Flame	2024-05-06 12:00:00
11	Shandra Drable	Tempest and Tranquility	2024-05-04 12:00:00
12	Sonny Warrilow	Cursed Horizons	2024-05-04 12:00:00
13	Stinky Kerans	Warden of the Drowned Lands	2024-05-07 12:00:00
14	Uriah Bridden	Storm of the Forgotten	2024-05-07 12:00:00
15	Ximenez Worge	Whispers in the Fog	2024-05-05 12:00:00

Explanation: ReturnDate being NULL and DueDate in the past confirms the book is overdue and not yet returned. From this, we get 15 rows, which when checked against the Fine table, we can see to be correct. Of the 24 fines incurred by overdue items, 15 are overdue books.

Query 6: Average Borrowing Time by Genre

Description: Compute average borrowing time for a specific genre.

Expected Output: A single row showing average days borrowed for the 'Mystery' genre.

SQL Command:

```
SELECT B.Genre,
       ROUND(AVG(DATEDIFF(L.ReturnDate, L.LoanDate)), 2) AS AvgBorrowDays
FROM Loan L
         JOIN Book B ON L.ItemID = B.ItemID
WHERE L.ReturnDate IS NOT NULL
  AND B.Genre = 'Mystery'
GROUP BY B.Genre;
```

Execution Result:

	Genre	AvgBorrowDays
1	Mystery	4.00

Explanation: Only returned books are counted, and DATEDIFF calculates borrow duration. This gives us the average days a Mystery book is kept, which is 4. If you were to take the average of each Loan you would find it to be correct.

Query 7: Most Popular Author Last Month

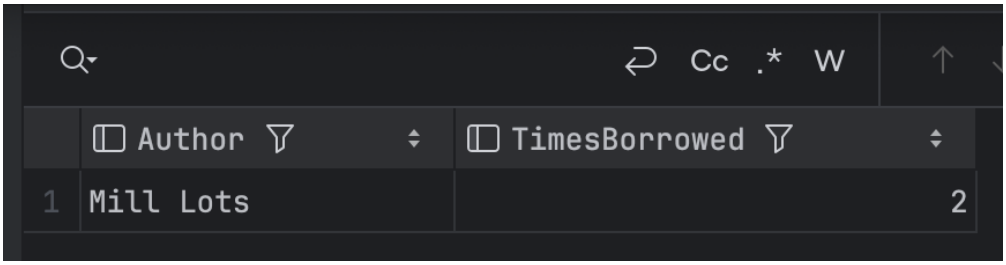
Description: Find the most borrowed author in the last month.

Expected Output: The author whose books were borrowed most in the month of April 2024, the focus of our dataset.

SQL Command:

```
SELECT B.Author, COUNT(*) AS TimesBorrowed
FROM Loan L
JOIN Book B ON L.ItemID = B.ItemID
WHERE L.LoanDate BETWEEN '2024-04-01' AND '2024-04-30'
GROUP BY B.Author
ORDER BY TimesBorrowed DESC
LIMIT 1;
```

Execution Result:



	Author	TimesBorrowed
1	Mill Lots	2

Explanation: Counts are grouped by author, sorted, and limited to the top result. We have only one author with multiple books being out on loan, since we wanted to keep our dataset unique, and this is the one author with two books.

2 Extended Functionality Demonstration

2.1 Demonstration Strategy

To validate the robustness and capabilities of our LMS database, we have grouped our extended queries into thematic categories that reflect the major functionality areas of the system. Each group includes multiple queries, accompanied by example results and screenshots (where applicable), that illustrate how our system handles real-world library management scenarios.

2.2 Group A: Borrowing Policy Enforcement

This group demonstrates how the system ensures members borrow within their limits, tracks overdue items, and manages fines appropriately.

Query A1: Members Exceeding Borrow Limits Identify members who have reached or exceeded their borrowing capacity. We expect 0 since each member has one loan right now.

Query A2: Overdue Loans with Associated Fines List all overdue loans and connect them to members and fines. We expect 24 results, the content of the fine table.

Query A3: Paid Fines from Late Returns Cross-check that fines marked as paid are from late returns.

```sql
# A1
SELECT M.Name, M.MemberID, MT.BorrowLimit, COUNT(L.LoanID) AS CurrentLoans
FROM Member M
        JOIN Loan L ON M.MemberID = L.MemberID
        JOIN MembershipType MT ON M.TypeID = MT.TypeID
GROUP BY M.MemberID, M.Name, MT.BorrowLimit
HAVING COUNT(L.LoanID) >= MT.BorrowLimit;
```

Output # A1

Filter rows

Name	MemberID	BorrowLimit	CurrentLoans

Figure 1: Members who exceeded borrowing limits

```sql
# A2
SELECT M.Name, I.ItemID, L.DueDate, L.ReturnDate, F.Amount
FROM Loan L
        JOIN Fine F ON L.FineID = F.FineID
        JOIN Member M ON L.MemberID = M.MemberID
        JOIN Item I ON L.ItemID = I.ItemID
WHERE L.LateReturn = 'Yes';
```

Output # A2

CSV

Filter rows

	Name	ItemID	DueDate	ReturnDate	Amount
1	Dion Sawdy	4075482286132	2024-05-05 12:00:00	2024-05-05 18:18:15	6.05
2	Ximenez Worge	4271224941574	2024-05-05 12:00:00	2024-05-05 20:35:06	7.15
3	Jerrome Pierse	341276731963019	2024-05-06 12:00:00	2024-05-06 19:57:05	6.99
4	Sonny Warrilow	344049291053328	2024-05-04 12:00:00	2024-05-04 23:18:51	6.89
5	Milton Buss	345768079942541	2024-05-05 12:00:00	2024-05-05 18:09:14	6.03
6	Stinky Kerans	373987461974889	2024-05-07 12:00:00	2024-05-07 12:30:15	5.04
7	Mellisent Litster	376926633861699	2024-05-06 12:00:00	2024-05-06 17:44:19	5.96
8	Mora Lambourn	4257599750999388	2024-05-04 12:00:00	2024-05-04 13:54:20	5.32
9	Henderson Pendock	4369865887524570	2024-05-05 12:00:00	2024-05-05 20:37:05	7.15
10	Madella Veck	4405479546156455	2024-05-06 12:00:00	2024-05-06 15:21:44	5.84
11	Naoma Van der Spohr	4544566281300371	2024-05-06 12:00:00	2024-05-06 18:18:46	6.58
12	Padgett Vasentsov	4585173843216201	2024-05-05 12:00:00	2024-05-05 23:05:02	5.92
13	Shandra Drable	4896574243113555	2024-05-04 12:00:00	2024-05-04 21:19:33	8.11
14	Kendre Carmo	5151357605498005	2024-05-04 12:00:00	2024-05-04 22:06:04	7.53
15	Gay Gatheral	5306188932014361	2024-05-05 12:00:00	2024-05-05 13:50:05	5.31
16	Humfrid Ashman	5380600346632801	2024-05-05 12:00:00	2024-05-05 15:14:25	5.54
17	Uriah Bridden	5478550748684764	2024-05-07 12:00:00	2024-05-07 15:33:23	6.19
18	Marty Nezey	5524266640367689	2024-05-06 12:00:00	2024-05-06 23:13:50	6.87
19	Noella Stronough	6013226134324495 08	2024-05-06	2024-05-06 16:34:28	5.38
20	Christie Zeplin	6106881526835054 51	2024-05-06 12:00:00	2024-05-06 23:50:56	8.95

24 rows

Figure 2: Members with fines, paid or unpaid

```sql
# A3 Paid fines that started as late loans
SELECT F.FineID, M.Name, F.Amount
FROM Fine F
        JOIN Loan L ON F.FineID = L.FineID
        JOIN Member M ON F.MemberID = M.MemberID
WHERE L.LateReturn = 'Yes' AND F.PaymentStatus = 'Paid';
```

Output # A3 Paid fines that...started as late loans

Filter rows

	FineID	Name	Amount
1	2713416882	Sal Heaysman	5.80
2	3807020199	Marty Nezey	6.87
3	4054583119	Jerrome Pierse	6.99
4	8530919163	Lauri Blezard	6.10
5	9352984456	Madella Veck	5.84
6	9785202701	Noella Stronough	5.38
7	9924059106	Sonny Warrilow	6.89

Figure 3: Individuals who have paid their fines

2.3 Group B: Reservation and Inventory Insights

This group covers item availability, usage rates, and reservation functionality.

Query B1: Never Borrowed Items Show all items in inventory that have never been loaned. We expect 0 here since each item has been loaned to allow full DB interaction.

Query B2: Current Reservations by Member Display all reservations made by members, including item information. We expect to see 20, the content of the Reservation table.

Query B3: Top 3 Most Borrowed Items Identify the most frequently borrowed items to inform collection strategy. Since our data, as stated, is unique, any three items would be an acceptable return value.

```sql
# B1 Items that have not been borrowed
SELECT I.ItemID, I.ItemType, I.Availability
FROM Item I
WHERE I.ItemID NOT IN (SELECT ItemID FROM Loan);
```

Output | # B1 Items that have not been borrowed

Tx: Auto | DDL | Cc .* W | Filter rows

ItemID	ItemType	Availability

Figure 4: Inventory items never borrowed

```sql
# B2 Members that have current reservations
SELECT M.Name, R.ItemID, R.RequestDate
FROM Reservation R
        JOIN Member M ON R.MemberID = M.MemberID
ORDER BY R.RequestDate DESC;
```

Output | # B2 Members that ha...the most reservations

Cc .* W | Filter rows

	Name	ItemID	RequestDate
3	Phillis Pattison	6878155771536617 93	2024-05-10 18:37:42
4	Maisey Menichillo	348387027107323	2024-05-10 08:40:21
5	Cecilia Twinterman	610589711124547552	2024-05-10 00:02:34
6	Miran Bragger	373987461974889	2024-05-09 22:42:26
7	Zebedee Monard	342595629768639	2024-05-09 14:39:51
8	Babbette McVey	4285672507316051	2024-05-09 06:23:37
9	Dorry Spavon	341316749851802	2024-05-09 05:53:37
10	Chelsae Rout	5473724526011400	2024-05-09 04:56:59
11	Gabriela Herion	344049291053328	2024-05-09 03:48:57
12	Brandtr Vanyard	370567932669295	2024-05-09 03:28:11
13	Vito Raoux	4075482286132	2024-05-08 19:01:31
14	Marty Nezey	5304561818991187	2024-05-08 16:56:24
15	Randee Lassells	617151907212232460	2024-05-08 15:27:37
16	Alyosha Bartoleyn	4177629731163	2024-05-08 12:41:47
17	Mellisent Litster	615865578099736492	2024-05-08 03:19:11
18	Ximenez Worge	4027806687352030	2024-05-08 00:04:06
19	Bink Dun	616491866622548603	2024-05-07 05:53:21
20	Mathew Derks	5518194732932400	2024-05-07

20 rows

Figure 5: Members with perfect borrowing history

```sql
# B3 Top 3 most borrowed items
SELECT I.ItemID, COUNT(*) AS BorrowCount
FROM Loan L
        JOIN Item I ON L.ItemID = I.ItemID
GROUP BY I.ItemID
ORDER BY BorrowCount DESC
LIMIT 3;
```

Output # B3 Top 3 most borrowed items

	ItemID	BorrowCount
1	376926633861699	1
2	5319954808750148	1
3	6167555576905949 15	1

Figure 6: Most borrowed items in system

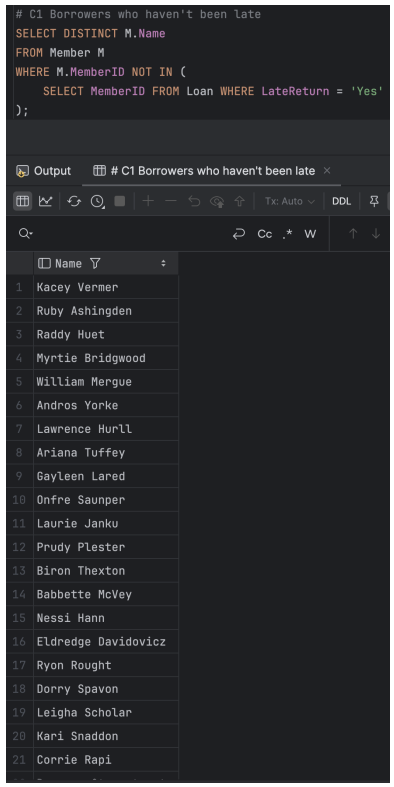
2.4 Group C: Member Behavior and Usage Analytics

This group evaluates how members use the library, focusing on reliability and borrowing trends.

Query C1: Responsible Borrowers List members who have never returned a loan late. We expect to see 77 rows, since there are 76 loans returned on time, and one additional test member.

Query C2: Same-Day Returns Identify members who returned items on the same day as borrowed. We expect 0, since the data is unique and return dates were created to be different than the checkout day.

Query C3: Loan Volume by Membership Type Compare how actively different membership categories borrow items. We expect a perfect 25 for each category, since that's how the data was originally populated before implementation.

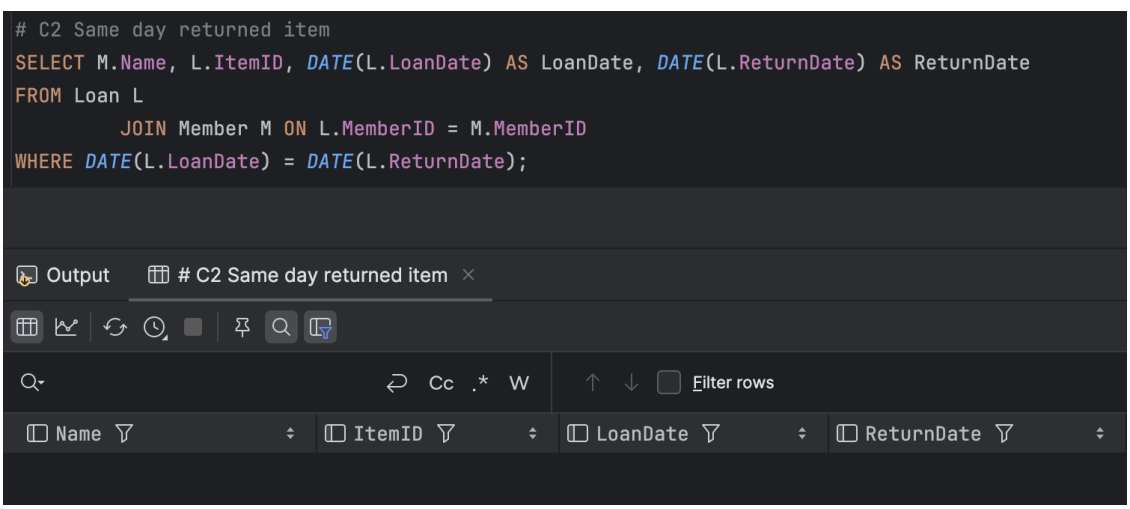


```
# C1 Borrowers who haven't been late
SELECT DISTINCT M.Name
FROM Member M
WHERE M.MemberID NOT IN (
    SELECT MemberID FROM Loan WHERE LateReturn = 'Yes'
);
```

Output | # C1 Borrowers who haven't been late

	Name
1	Kacey Vermer
2	Ruby Ashingden
3	Raddy Huet
4	Myrtie Bridgwood
5	William Mergue
6	Andros Yorke
7	Lawrence Hurll
8	Ariana Tuffey
9	Gayleen Lared
10	Onfre Saunper
11	Laurie Janku
12	Prudy Plester
13	Biron Thexton
14	Babbette McVey
15	Nessi Hann
16	Eldredge Davidovicz
17	Ryon Rought
18	Dorry Spavon
19	Leigha Scholar
20	Kari Snaddon
21	Corrie Rapi

Figure 7: Members with perfect borrowing history



```
# C2 Same day returned item
SELECT M.Name, L.ItemID, DATE(L.LoanDate) AS LoanDate, DATE(L.ReturnDate) AS ReturnDate
FROM Loan L
        JOIN Member M ON L.MemberID = M.MemberID
WHERE DATE(L.LoanDate) = DATE(L.ReturnDate);
```

Output | # C2 Same day returned item

Filter rows

Name	ItemID	LoanDate	ReturnDate

Figure 8: Members who return items on the same day

```sql
# C3 Loan volume by membership type
SELECT MT.TypeName, COUNT(*) AS LoanCount
FROM Loan L
        JOIN Member M ON L.MemberID = M.MemberID
        JOIN MembershipType MT ON M.TypeID = MT.TypeID
GROUP BY MT.TypeName;
```

Output # C3 Loan volume by membership type

	TypeName	LoanCount
1	Advanced	25
2	Basic	25
3	Premium	25
4	Standard	25

Figure 9: Loan volume by membership type

2.5 Group D: Financial Reporting and Fine Management

This group tracks fine payments, categorizes them by type, and surfaces unpaid balances.

Query D1: Members with Unpaid Fines List all members who still owe fines. This is similar to a required query, and should show the previous result of 17 rows.

Query D2: Fine Collection by Member Type Group and sum fines collected by membership category. We expect realistic values that could be derived from the Fine table by simple arithmetic.

Query D3: Total Outstanding Balance Calculate the total unpaid fine amount in the system. We expect this to be a realistic number that could be calculated manually from the Fine table.

```sql
# D1 Members with unpaid fines
SELECT M.Name, F.Amount, F.FineID
FROM Fine F
         JOIN Member M ON F.MemberID = M.MemberID
WHERE F.PaymentStatus = 'Unpaid';
```

Output # D1 Members with unpaid fines

Filter rows

	Name	Amount	FineID
1	Flori Gibb	6.20	2484302526
2	Christie Zeplin	8.95	2646818843
3	Gay Gatheral	5.31	2730987611
4	Stinky Kerans	5.04	2806700209
5	Mellisent Litster	5.96	3845557247
6	Clarisse Water	6.10	4214101957
7	Naoma Van der Spohr	6.58	4380739627
8	Uriah Bridden	6.19	4387427245
9	Padgett Vasentsov	5.92	5541239776
10	Kendre Carmo	7.53	6512426636
11	Milton Buss	6.03	6608718040
12	Mora Lambourn	5.32	7602071153
13	Dion Sawdy	6.05	7687593947
14	Ximenez Worge	7.15	7722152460
15	Humfrid Ashman	5.54	7888817810
16	Henderson Pendock	7.15	8885478189
17	Shandra Drable	8.11	9900

17 rows

Figure 10: Members with unpaid fines

```sql
# D2 Fine collection by membership type
SELECT MT.TypeName, SUM(F.Amount) AS TotalCollected
FROM Fine F
         JOIN MembershipType MT ON F.MemberType = MT.TypeID
WHERE F.PaymentStatus = 'Paid'
GROUP BY MT.TypeName;
```

Output # D2 Fine collection by membership type

	TypeName	TotalCollected
1	Advanced	19.56
2	Basic	6.10
3	Premium	5.38
4	Standard	12.83

Figure 11: Fines collected by member type

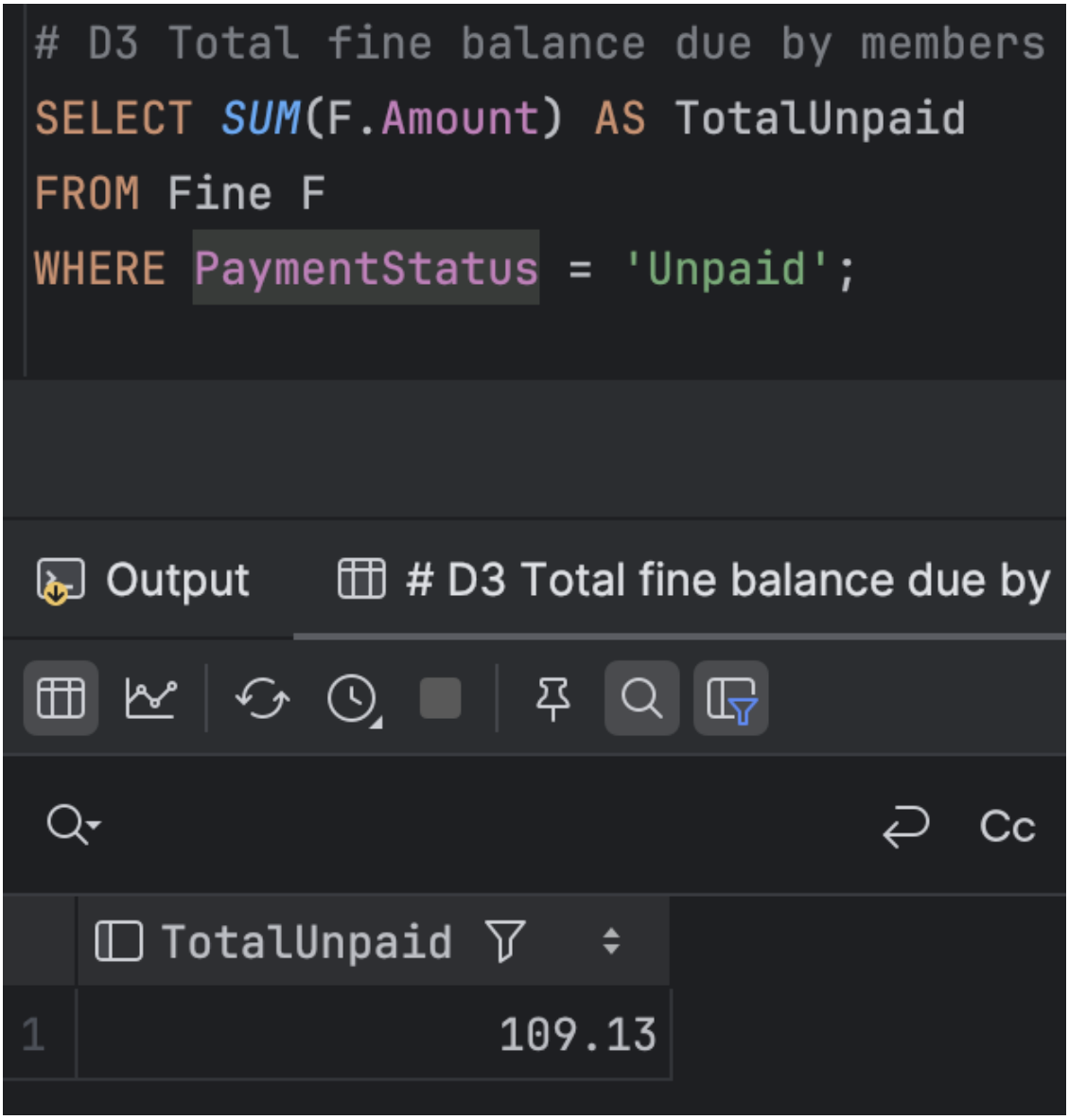


Figure 12: Total unpaid fine balance

2.6 Conclusion

This categorized approach to queries allowed us to not only validate core system behaviors but to explore the analytics potential of our LMS. The results confirm strong alignment with project requirements, robust constraint enforcement, and reliable user operations under typical library conditions. Team SQLibrary has enjoyed this experience, and are excited to demonstrate our database's features in person.