Phase 3: Develop Backend – More Complex Queries

Student#: 220274536
Sur (Family) Name: Markham
Given Name: Sienna

Student#: 219447192
Sur (Family) Name: Ryan
Given Name: Eamon

Student#: 220416038
Sur (Family) Name: Mollah
Given Name: Mahjabin

Section: C/Phase 3
Github Repository: https://github.com/eamontryan/DIGT_3107_team_3

Objective:

Build and test the backend of the system by implementing more complex SQL queries that reflect real-world operations. These queries will provide meaningful insights, support decision-making, and power the application's functionality.

Tasks:

Develop advanced queries that go beyond simple SELECT statements. Here are some query examples for each of the systems.

Query	Expected Output
1. List overdue books and associated fines	Book title, borrower, due date, days overdue, fine amount
2. Find most borrowed books	Book titles with total borrow count
3. Get borrowing history of a specific member	List of books, borrow/return dates
4. List books currently borrowed and due soon	Books due in the next 3 days with borrower info

5. Identify books never borrowed	Titles and IDs of books that are in stock but never issued
6. Show Availability and Circulation Status of a Specific Book	Number of available copies, Number of borrowed copies, Expected return dates for borrowed copies

---

Complex Queries

1. List overdue books and associated fines

```
SELECT
    b.title AS book_title,
    m.name AS borrower,
    l.due_date,
    DATEDIFF(CURDATE(), l.due_date) AS days_overdue,
    li.fine_amount
FROM loan_item li
JOIN loan l ON li.loan_id = l.loan_id
JOIN member m ON l.member_id = m.member_id
JOIN book b ON li.book_id = b.book_id
WHERE li.status = 'borrowed' AND l.due_date < CURDATE();
```

2. Find most borrowed books

```
SELECT
    b.title AS book_title,
    COUNT(li.book_id) AS total_borrow_count
FROM loan_item li
JOIN book b ON li.book_id = b.book_id
GROUP BY b.book_id
ORDER BY total_borrow_count DESC
LIMIT 10;
```

3. Get borrowing history of a specific member

```
SELECT
    b.title AS book_title,
    l.borrow_date,
    l.due_date,
    li.return_date,
    li.status
```

```
FROM loan l
JOIN loan_item li ON l.loan_id = li.loan_id
JOIN book b ON li.book_id = b.book_id
WHERE l.member_id = ?  -- replace ? with the specific member_id
ORDER BY l.borrow_date DESC;
```

4. List books currently borrowed and due soon (3 days)

```
SELECT
   b.title AS book_title,
   m.name AS borrower,
   l.due_date
FROM loan_item li
JOIN loan l ON li.loan_id = l.loan_id
JOIN member m ON l.member_id = m.member_id
JOIN book b ON li.book_id = b.book_id
WHERE li.status = 'borrowed'
  AND l.due_date BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL
3 DAY)
ORDER BY l.due_date;
```

5. Identify books never borrowed

```
SELECT
   b.book_id,
   b.title
FROM book b
LEFT JOIN loan_item li ON b.book_id = li.book_id
WHERE li.book_id IS NULL;
```

6. Show availability and circulation status of a specific book

```
SELECT
   b.title,
   b.total_copies,
   b.available_copies,
   (b.total_copies - b.available_copies) AS borrowed_copies,
   GROUP_CONCAT(DISTINCT l.due_date ORDER BY l.due_date SEPARATOR ', ')
AS expected_return_dates
FROM book b
LEFT JOIN loan_item li ON b.book_id = li.book_id AND li.status = 'borrowed'
LEFT JOIN loan l ON li.loan_id = l.loan_id
```

```
WHERE b.book_id = ?  -- replace ? with the book ID
GROUP BY b.book_id;
```

7. Get total fines per member

```
SELECT
    m.name AS member_name,
    SUM(li.fine_amount) AS total_fines
FROM loan_item li
JOIN loan l ON li.loan_id = l.loan_id
JOIN member m ON l.member_id = m.member_id
GROUP BY m.member_id
HAVING total_fines > 0
ORDER BY total_fines DESC;
```

8. Find members who currently have overdue books

```
SELECT DISTINCT
    m.member_id,
    m.name,
    m.phone_number
FROM member m
JOIN loan l ON m.member_id = l.member_id
JOIN loan_item li ON l.loan_id = li.loan_id
WHERE li.status = 'borrowed' AND l.due_date < CURDATE();
```

9. List books tagged under a specific category with their tags

```
SELECT
    b.title AS book_title,
    c.category_name,
    GROUP_CONCAT(t.tag_name SEPARATOR ', ') AS tags
FROM book b
LEFT JOIN category c ON b.category_id = c.category_id
LEFT JOIN book_tag bt ON b.book_id = bt.book_id
LEFT JOIN tag t ON bt.tag_id = t.tag_id
WHERE c.category_name = ?  -- e.g. 'Science Fiction'
GROUP BY b.book_id;
```

1) List overdue books and associated fines

Purpose:
This query retrieves all books that are currently overdue and calculates the number of days past the due date, along with the associated fines. It joins the loan, loan_item, member, and book tables to display relevant borrower and book details. The query filters only active borrowed items where the due date has passed (due_date < CURDATE()), and uses DATEDIFF to compute the total days overdue. This information supports overdue notifications, fine assessment, and library account management.

Query results Screenshot:

```sql
-- Query 1: List overdue books and associated fines

SELECT
    b.title AS book_title,
    m.name AS borrower,
    l.due_date,
    DATEDIFF(CURDATE(), l.due_date) AS days_overdue,
    li.fine_amount
FROM loan_item li
JOIN loan l ON li.loan_id = l.loan_id
JOIN member m ON l.member_id = m.member_id
JOIN book b ON li.book_id = b.book_id
WHERE li.status = 'borrowed' AND l.due_date < CURDATE();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_title	borrower	due_date	days_overdue	fine_amount
Alien Encounters	Laura Smith	2025-09-19	44	430.00
The Lost Universe	Laura Smith	2025-09-19	44	430.00
The Ancient World	Daniel Kim	2025-09-26	37	360.00
The Renaissance	Frank Wilson	2025-10-02	31	300.00
Revolutionary Leaders	Grace Miller	2025-10-04	29	280.00
Historical Wonders	Henry Clark	2025-10-09	24	230.00
Pride and Prejudice	Henry Clark	2025-10-09	24	230.00

---

2) Find most borrowed books

Purpose:
This query identifies the most frequently borrowed books by counting how many times each title appears in the loan records. It aggregates borrow activity from the loan_item table by book_id, joins with the book table to display titles, and orders the results in descending order of total borrow count. The top results provide insight into high-demand materials, which supports collection development, budgeting, and library display decisions.

Query results Screenshot:

```sql
-- Query 2: Find most borrowed books

SELECT
    b.title AS book_title,
    COUNT(li.book_id) AS total_borrow_count
FROM loan_item li
JOIN book b ON li.book_id = b.book_id
GROUP BY b.book_id
ORDER BY total_borrow_count DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

book_title	total_borrow_count
The Galactic Journey	1
Time and Space	1
Alien Encounters	1
The Lost Universe	1
Mars Colonies	1
The Ancient World	1
World Wars	1
The Renaissance	1
Revolutionary Leaders	1
Historical Wonders	1

---

3) Get borrowing history of a specific member

Purpose:
This query retrieves the complete borrowing history for a specific library member, including book titles, borrow dates, due dates, return dates, and the current loan status. It filters results using the member's unique identifier (member_id) and orders records in descending order by borrow date to show the most recent activity first. This information is useful for circulation desk support, patron account inquiries, and resolving disputes regarding overdue or returned materials.

Query results Screenshot:

```sql
-- Query 3: Get borrowing history of a specific member

SELECT
    b.title AS book_title,
    l.borrow_date,
    l.due_date,
    li.return_date,
    li.status
FROM loan l
JOIN loan_item li ON l.loan_id = li.loan_id
JOIN book b ON li.book_id = b.book_id
WHERE l.member_id = 1   -- change member_id here if needed
ORDER BY l.borrow_date DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_title	borrow_date	due_date	return_date	status
The Galactic Journey	2025-09-01	2025-09-15	2025-09-10	returned
Time and Space	2025-09-01	2025-09-15	2025-09-13	returned

---

4) List books currently borrowed and due soon (3 days)

Purpose:
This query identifies books that are currently checked out and approaching their due date within the next three days. It filters active loans by selecting records where the loan status is 'borrowed' and the due date falls between the current date and three days ahead. By listing upcoming due returns, this query helps library staff send timely reminders and manage circulation workflows effectively, reducing the likelihood of overdue items.

Query results Screenshot:

```sql
-- Query 4: List books currently borrowed and due soon (3 days)

SELECT
    b.title AS book_title,
    m.name AS borrower,
    l.due_date
FROM loan_item li
JOIN loan l ON li.loan_id = l.loan_id
JOIN member m ON l.member_id = m.member_id
JOIN book b ON li.book_id = b.book_id
WHERE li.status = 'borrowed'
  AND l.due_date BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 3 DAY)
ORDER BY l.due_date;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_title	borrower	due_date

---

5) Identify books never borrowed

Purpose:
This query identifies books in the catalog that have never been checked out by performing a LEFT JOIN between the book and loan_item tables and selecting titles with no corresponding loan records. By highlighting materials that have never circulated, the library can evaluate low-use items for potential weeding, re-cataloging, or targeted promotion to improve collection performance and shelf efficiency.

Query results Screenshot:

```sql
-- Query 5: Identify books never borrowed

SELECT
    b.book_id,
    b.title
FROM book b
LEFT JOIN loan_item li ON b.book_id = li.book_id
WHERE li.book_id IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_id	title
12	The Great Gatsby
13	To Kill a Mockingbird
14	1984
15	Moby Dick

---

6) Show availability & circulation status of a specific book

Purpose:
This query provides a detailed availability and circulation summary for a specific book. It displays the total number of copies, how many are currently available, and calculates the number of borrowed copies by subtracting available copies from the total. It also retrieves expected return dates for all active loans using GROUP_CONCAT, allowing staff and patrons to see when checked-out copies are due back. This information supports hold requests, collection management, and real-time inventory visibility.

Query results Screenshot:

```sql
-- Query 6: Show availability and circulation status of a specific book

SELECT
    b.title,
    b.total_copies,
    b.available_copies,
    (b.total_copies - b.available_copies) AS borrowed_copies,
    GROUP_CONCAT(DISTINCT l.due_date ORDER BY l.due_date SEPARATOR ', ') AS expected_return_dates
FROM book b
LEFT JOIN loan_item li ON b.book_id = li.book_id AND li.status = 'borrowed'
LEFT JOIN loan l ON li.loan_id = l.loan_id
WHERE b.book_id = 1    -- change book_id here as needed
GROUP BY b.book_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	total_copies	available_copies	borrowed_copies	expected_return_dates
The Galactic Journey	5	3	2	NULL

---

7) Get total fines per member

Purpose:
This query calculates the total outstanding fines owed by each library member by summing all recorded fine amounts associated with their loan items. It groups results by member and uses a HAVING clause to include only patrons with a balance greater than zero. This report assists staff with fine management, payment processing, and account follow-up at checkout or when reviewing borrower account status.

Query results Screenshot:

```sql
-- Query 7: Get total fines per member

SELECT
    m.name AS member_name,
    SUM(li.fine_amount) AS total_fines
FROM loan_item li
JOIN loan l ON li.loan_id = l.loan_id
JOIN member m ON l.member_id = m.member_id
GROUP BY m.member_id
HAVING total_fines > 0
ORDER BY total_fines DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

member_name	total_fines
Laura Smith	860.00
Henry Clark	460.00
Daniel Kim	360.00
Frank Wilson	300.00
Grace Miller	280.00
Catherine Lee	20.00

---

8) Find members who currently have overdue books

Purpose:
This query identifies library members who currently have overdue books by selecting patrons with active loan items (status = 'borrowed') where the due date has already passed. It returns distinct member records to avoid duplicates and provides contact information for follow-up. This report supports overdue reminders, account notifications, and proactive circulation management.

Query results Screenshot:

```sql
-- Query 8: Find members who currently have overdue books

SELECT DISTINCT
    m.member_id,
    m.name,
    m.phone_number
FROM member m
JOIN loan l ON m.member_id = l.member_id
JOIN loan_item li ON l.loan_id = li.loan_id
WHERE li.status = 'borrowed'
  AND l.due_date < CURDATE();
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

member_id	name	phone_number
2	Laura Smith	4165552222
4	Daniel Kim	4165554444
6	Frank Wilson	4165556666
7	Grace Miller	4165557777
8	Henry Clark	4165558888

---

9) List books under a specific category with their tags

Purpose:
This query retrieves all books belonging to a specified category and displays their associated tags in a single, readable list. It joins the book, category, book_tag, and tag tables, and uses GROUP_CONCAT to combine multiple tags for each book. Because the query uses left joins, books without assigned tags are still included in the results. This output supports browsing features, category-based recommendations, and improved catalog navigation for patrons.

Query results Screenshot:

```sql
-- Query 9: List books tagged under a specific category with their tags

SELECT
    b.title AS book_title,
    c.category_name,
    GROUP_CONCAT(t.tag_name SEPARATOR ', ') AS tags
FROM book b
LEFT JOIN category c ON b.category_id = c.category_id
LEFT JOIN book_tag bt ON b.book_id = bt.book_id
LEFT JOIN tag t ON bt.tag_id = t.tag_id
WHERE c.category_name = 'Science Fiction'   -- change category if needed
GROUP BY b.book_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_title	category_name	tags
The Galactic Journey	Science Fiction	Space Travel, Adventure, Exploration
Time and Space	Science Fiction	Time Travel, Quantum Physics, Paradox
Alien Encounters	Science Fiction	Exploration, Extraterrestrial Life, UFOs
The Lost Universe	Science Fiction	Exploration, Parallel Worlds, Fantasy
Mars Colonies	Science Fiction	Exploration, Colonization, Mars

---

Deliverables for Phase 3:

1. SQL script with complex queries (saved in .sql)
2. Provide sample data for testing queries
3. At least 6–8 complex SQL queries
4. Create team#_phase3.pdf containing:
    a. Query results (screenshots)
    b. A short report explaining each query's purpose and expected use

Submission Instructions:

Upload team#_phase3.pdf and other .sql files to your GitHub repository and then post your GitHub repository link on eClass.

Must Have:

Include this information for all team members as a comment in every .sql file that you submit.

Student#:

Sur (Family) Name:

Given Name:

Section: