Project Demonstration

The DataBasers™

1. Database Usage

1.1 Access

To access the database directly, first log in to the EECS Cycle Servers using a valid set of credentials. Then, access the MariaDB database using the following command:

```
mysql -h mysql.eecs.ku.edu -u 447s25_j776w781 -p
```

When the terminal prompts for a password, enter "**ohN7iewa**". Then, to navigate to the project database, execute the following command:

```
use 447s25_j776w781;
```

Executing these commands yields direct access to the project database.

1.2 Sample Queries

Below is a selection of queries outlined in the Database Requirements document, the SQL command to implement the query, and the results of each query:

List all books by a specific author:

Using J.R.R. Tolkien as the author.

SQL Command:

```
select itemID, title, ISBN, author from book natural join media where author like "%Tolkien";
```

Output:

```
+--------+-----------+---------------+----------------+
| itemID | title     | ISBN          | author         |
+--------+-----------+---------------+----------------+
|     13 | The Hobbit | 9780547928227 | J.R.R. Tolkien |
|     61 | The Hobbit | 9780547928227 | J.R.R. Tolkien |
|     62 | The Hobbit | 9780547928227 | J.R.R. Tolkien |
|     64 | The Hobbit | 9780547928227 | J.R.R. Tolkien |
+--------+-----------+---------------+----------------+
4 rows in set (0.000 sec)
```

Find books by publication year:

Using 600 as a sample publication year.

SQL Command:

```
select * from book natural join media where publicationYear = 600;
```

Output:

```
+-------+--------------+---------------+-------+-------------+---------+---------------+------------------+--------------+-------------------+
| itemID | ISBN         | author        | genre | title       | itemType | publicationYear | availabilityStatus | specialPremium | specialRestriction |
+-------+--------------+---------------+-------+-------------+---------+---------------+------------------+--------------+-------------------+
|     66 | 9999999999999 | Abdul Alhazred | Gothic | Necronomicon | book     |            600 | unavailable       |             7 | rare              |
+-------+--------------+---------------+-------+-------------+---------+---------------+------------------+--------------+-------------------+
1 row in set (0.000 sec)
```

Check membership status:

Using 25 as a sample userID.

SQL Command:

```
select * from member natural join user where userID=25;
```

Output:

userID	typeNAME	borrowingLimit	lateFeeRate	name	phoneNumber	emailAddress	physicalAddress	userType	accountStatus
25	regular	5	4	LeBron James	111-111-1111	kingjames@email.com	1234 King St Raleigh NC 27601	member	active

1 row in set (0.001 sec)

Fine calculation:

SQL Command:

```
select userID, name, sum(amount) as "Total Fees" from (member natural
join user) left join fine on userID=memberID group by userID, name;
```

Output:

userID	name	Total Fees
1	Alham Aihmoud	NULL
2	Dalan Dyvis	NULL
3	Woshua Jelicky	NULL
4	Aldley Ashdave	NULL
5	Ahyyir Named	121
6	Alabama Almood	NULL
7	Nate Ahmed	NULL
8	Nason Gibthan	2
9	John Chumbles	NULL
10	Frank Gill	NULL
11	Gertrude Mollychalk	NULL
12	TingHao Guo	NULL
13	Ahkariya Zamed	NULL
14	Pidur Vandiripally	18
15	Dunchan Lin	NULL
16	Adam Nedal	8
17	Dongo Roja	NULL
18	China Walker	NULL
19	Connie Harris	NULL
20	Zaatar Mustafa	NULL
21	Bowen Maresh	NULL
22	Adhman Praikari	NULL
23	Puke Maresh	NULL
24	Mahtab Sheshappa	NULL
25	LeBron James	NULL
26	Emily Carter	NULL
27	Liam Rodriguez	40
28	Sophia Patel	NULL
29	Noah Thompson	NULL
30	Ava Martinez	1
31	Benjamin Lee	16
32	Mia Nguyen	NULL
33	Elijah Walker	NULL
34	Isabella Kim	NULL
35	James Scott	10
36	Charlotte Davis	NULL
37	William Young	NULL
38	Amelia Harris	NULL
39	Lucas Hall	NULL
40	Harper Robinson	5
41	Henry Lewis	NULL
42	Evelyn Allen	NULL
43	Alexander Wright	12
44	Abigail King	NULL
45	Daniel Green	NULL
46	Scarlett Baker	NULL
47	Jackson Perez	NULL
48	Ella Adams	NULL
49	Michael Rivera	NULL
50	Grace Torres	NULL

50 rows in set (0.000 sec)

Book availability by genre:

Using Fiction as a sample genre.

SQL command:

```
select * from book natural join media where genre like "%Fiction" and availabilityStatus="available";
```

Output:

itemID	ISBN	author	genre	title	itemType	publicationYear	availabilityStatus	specialPremium	specialRestriction
3	780679728757	Cormac McCarthy	Historical Fiction	Blood Meridian	book	1985	available	0	common
6	9781250773029	Orson Scott Card	Science Fiction	Ender's Game	book	1985	available	0	common
7	9781400079988	Leo Tolstoy	Historical Fiction	War and Peace	book	1865	available	0	common
8	9780811204811	Osamu Dazai	Psychological Fiction	No Longer Human	book	1948	available	0	common
16	9780307387899	Cormac McCarthy	Fiction	The Road	book	2006	available	0	common
17	9780143106586	Joseph Conrad	Historical Fiction	Heart of Darkness	book	1899	available	1	rare
63	9780307387899	Cormac McCarthy	Fiction	The Road	book	2006	available	0	common

Frequent borrowers of a specific genre:

Using “Fiction” as the sample genre.

SQL Command:

```
SELECT m.userID, COUNT(l.loanID) AS loan_count
FROM member m
JOIN loan l ON m.userID = l.memberID
JOIN book b ON l.itemID = b.itemID
WHERE b.genre LIKE '%Fiction'
GROUP BY m.userID
HAVING loan_count >= (
   SELECT MAX(loan_count)
   FROM (
        SELECT COUNT(l2.loanID) AS loan_count
        FROM member m2
        JOIN loan l2 ON m2.userID = l2.memberID
        JOIN book b2 ON l2.itemID = b2.itemID
        WHERE b2.genre LIKE '%Fiction'
        GROUP BY m2.userID
   ) AS user_loan_counts
);
```

Output:

userID	loan_count
7	2
31	2

2 rows in set (0.001 sec)

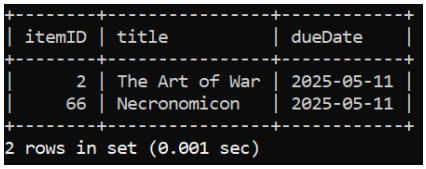
Books due within the next week:

Using May 9th, 2025, as the sample date.

SQL Command:

```
select itemID, title, dueDate from loan natural join book natural join media where returnDate is NULL and dueDate - DATE("2025-05-09") <= 7 order by dueDate;
```

Output:



```
+--------+---------------+------------+
| itemID | title         | dueDate    |
+--------+---------------+------------+
|      2 | The Art of War | 2025-05-11 |
|     66 | Necronomicon   | 2025-05-11 |
+--------+---------------+------------+
2 rows in set (0.001 sec)
```

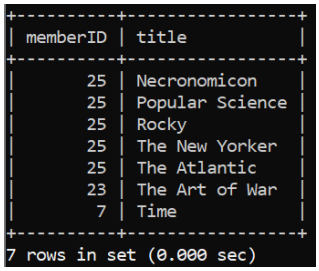
Members with overdue books:

Using May 12th, 2025, as the sample date, since all active loans will be overdue at this point.

SQL Command:

```
select memberID, title from loan natural join media where returnDate is NULL and dueDate < DATE("2025-05-12");
```

Output:



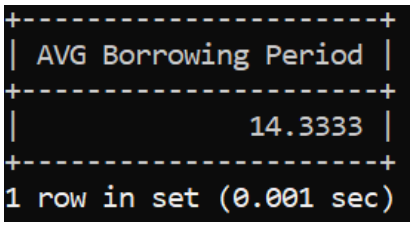
```
+----------+----------------+
| memberID | title          |
+----------+----------------+
|       25 | Necronomicon   |
|       25 | Popular Science |
|       25 | Rocky          |
|       25 | The New Yorker |
|       25 | The Atlantic   |
|       23 | The Art of War |
|        7 | Time           |
+----------+----------------+
7 rows in set (0.000 sec)
```

Average borrowing time in days for a given genre:

SQL command:

```
select AVG(DATEDIFF(returnDate, checkoutDate)) as "AVG Borrowing Period" from loan natural join book where genre like "%Fiction";
```

Output:



```
+----------------------+
| AVG Borrowing Period |
+----------------------+
|              14.3333 |
+----------------------+
1 row in set (0.001 sec)
```

Most borrowed author in the last month:

SQL command:

```
SELECT b.author, COUNT(*) AS borrow_count
FROM loan l
JOIN book b ON l.itemID = b.itemID
WHERE MONTH(l.checkoutDate) = MONTH(CURDATE() - INTERVAL 1 MONTH)
  AND YEAR(l.checkoutDate) = YEAR(CURDATE() - INTERVAL 1 MONTH)
GROUP BY b.author
ORDER BY borrow_count DESC;
```

Output:

author	borrow_count
Abdul Alhazred	1
Sun Tzu	1
J.R.R. Tolkien	1
Edgar Allan Poe	1

4 rows in set (0.001 sec)

Monthly fees reported by member type:

Including both paid and unpaid fines.

SQL command:

```
SELECT
    DATE_FORMAT(f.issueDate, '%Y-%m') AS month,
    m.typeNAME AS member_type,
    SUM(f.amount) AS total_fines
FROM fine f
JOIN member m ON f.memberID = m.userID
GROUP BY month, member_type
ORDER BY month, member_type;
```

Output:

month	member_type	total_fines
2022-01	senior	12
2022-02	regular	24
2022-02	student	41
2022-02	senior	3
2022-03	regular	17
2022-03	senior	18
2025-05	senior	118

7 rows in set (0.001 sec)

Exceeded borrowing limits:

SQL command:

```
SELECT
    m.userID,
    m.typeNAME,
    m.borrowingLimit,
    COUNT(b.loanID) AS current_borrowed
FROM member m
JOIN borrows b ON m.userID = b.memberID
GROUP BY m.userID, m.typeNAME, m.borrowingLimit
HAVING current_borrowed > m.borrowingLimit;
```

Output:

Empty set (0.001 sec)

Frequently borrowed items by member type:

SQL command:

```
SELECT
    ranked.member_type,
    ranked.itemID,
    me.title,
    ranked.borrow_count
FROM (
    SELECT
        m.typeNAME AS member_type,
        l.itemID,
        COUNT(*) AS borrow_count,
        RANK() OVER (
            PARTITION BY m.typeNAME
            ORDER BY COUNT(*) DESC
        ) AS rank
    FROM
        member m
    JOIN
        loan l ON m.userID = l.memberID
    GROUP BY
        m.typeNAME, l.itemID
) ranked
JOIN media me ON ranked.itemID = me.itemID
WHERE ranked.rank = 1
ORDER BY ranked.member_type, ranked.itemID;
```

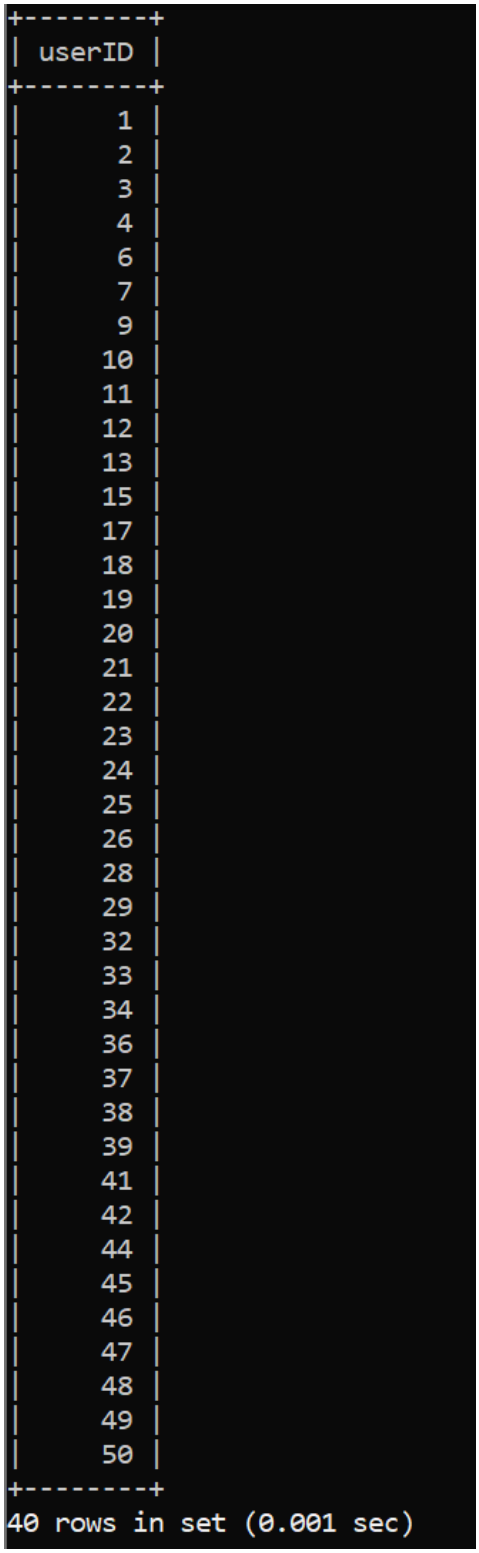
Output:

```
+-------------+--------+----------------------------------------+--------------+
| member_type | itemID | title                                  | borrow_count |
+-------------+--------+----------------------------------------+--------------+
| regular     |     59 | Only Murders in the Building: Season 3 |            3 |
| student     |     13 | The Hobbit                             |            2 |
| student     |     43 | Psycho                                 |            2 |
| student     |     56 | The Wire: Season 5                     |            2 |
| senior      |      2 | The Art of War                         |            4 |
+-------------+--------+----------------------------------------+--------------+
5 rows in set (0.001 sec)
```

Users who have never returned an item late:

```
SELECT DISTINCT u.userID
FROM user u
JOIN member m ON u.userID = m.userID
WHERE u.userID NOT IN (
      SELECT l.memberID
      FROM loan l
      WHERE l.returnDate > l.dueDate
)
AND u.accountStatus = 'active';
```

Output:



```
+--------+
| userID |
+--------+
|      1 |
|      2 |
|      3 |
|      4 |
|      6 |
|      7 |
|      9 |
|     10 |
|     11 |
|     12 |
|     13 |
|     15 |
|     17 |
|     18 |
|     19 |
|     20 |
|     21 |
|     22 |
|     23 |
|     24 |
|     25 |
|     26 |
|     28 |
|     29 |
|     32 |
|     33 |
|     34 |
|     36 |
|     37 |
|     38 |
|     39 |
|     41 |
|     42 |
|     44 |
|     45 |
|     46 |
|     47 |
|     48 |
|     49 |
|     50 |
+--------+
40 rows in set (0.001 sec)
```

Average loan duration:

```
SELECT
AVG(DATEDIFF(
      CASE
            WHEN l.returnDate IS NULL THEN CURRENT_DATE
            ELSE l.returnDate
      END,
      l.checkoutDate
)) AS average_loan_duration_days
FROM loan l
WHERE l.checkoutDate IS NOT NULL;
```

Output:

```
+-----------------------------+
| average_loan_duration_days |
+-----------------------------+
|                     14.7000 |
+-----------------------------+
1 row in set (0.000 sec)
```

1.3 Report Generation

Below is a selection of reports outlined in the Database Requirements document, the SQL commands to implement the report, and the results of each report:

Monthly Summary Report:

Total number of items loaned and total fees collected

SQL Query:

```
SELECT
    DATE_FORMAT(l.checkoutDate, '%Y-%m') AS month,
    COUNT(l.loanID) AS total_items_loaned,
    SUM(COALESCE(p.amount, 0)) AS total_fees_collected
FROM loan l
LEFT JOIN payment p ON DATE_FORMAT(l.checkoutDate, '%Y-%m') =
DATE_FORMAT(p.paymentDate, '%Y-%m')
WHERE DATE_FORMAT(l.checkoutDate, '%Y-%m') = DATE_FORMAT(CURRENT_DATE,
'%Y-%m')
GROUP BY DATE_FORMAT(l.checkoutDate, '%Y-%m');
```

Output: For the month of April

```
+---------+-------------------+---------------------+
| month   | total_items_loaned | total_fees_collected |
+---------+-------------------+---------------------+
| 2025-04 |                 9 |                   0 |
+---------+-------------------+---------------------+
1 row in set (0.001 sec)
```

Most popular items for the month

SQL Query:

```
SELECT
    m.title AS item_title,
    m.itemType AS media_type,
    COUNT(l.itemID) AS loan_count
FROM loan l
JOIN media m ON l.itemID = m.itemID
WHERE DATE_FORMAT(l.checkoutDate, '%Y-%m') = DATE_FORMAT(CURRENT_DATE,
'%Y-%m')
GROUP BY m.itemID, m.title, m.itemType
ORDER BY loan_count DESC
LIMIT 10;
```

Output: For the month of April

```
+--------------------+-----------+-----------+
| item_title         | media_type | loan_count |
+--------------------+-----------+-----------+
| The Hobbit         | book       |          1 |
| The New Yorker     | magazine   |          1 |
| Necronomicon       | book       |          1 |
| Time               | magazine   |          1 |
| The Art of War     | book       |          1 |
| Rocky              | digital    |          1 |
| The Tell-Tale Heart | book      |          1 |
| The Atlantic       | magazine   |          1 |
| Popular Science    | magazine   |          1 |
+--------------------+-----------+-----------+
9 rows in set (0.001 sec)
```

Statistics Breakdown:

SQL Query:

```
SELECT
    DATE_FORMAT(l.checkoutDate, '%Y-%m') AS month,
    m.typeNAME AS member_type,
    med.itemType AS media_type,
    COUNT(l.loanID) AS total_loans,
    AVG(DATEDIFF(COALESCE(l.returnDate, CURRENT_DATE), l.checkoutDate))
AS avg_loan_duration,
    SUM(COALESCE(l.lateFeeCharge, 0)) AS total_late_fees,
    COUNT(CASE WHEN l.returnDate > l.dueDate THEN 1 END) AS late_returns
FROM loan l
JOIN member m ON l.memberID = m.userID
JOIN media med ON l.itemID = med.itemID
WHERE DATE_FORMAT(l.checkoutDate, '%Y-%m') = DATE_FORMAT(CURRENT_DATE,
'%Y-%m')
GROUP BY
    DATE_FORMAT(l.checkoutDate, '%Y-%m'),
    m.typeNAME,
```

```
    med.itemType
ORDER BY
    month,
    member_type,
    media_type;
```

Output: For the month of April

month	member_type	media_type	total_loans	avg_loan_duration	total_late_fees	late_returns
2025-04	regular	book	1	7.0000	0	0
2025-04	regular	digital	1	7.0000	0	0
2025-04	regular	magazine	4	8.7500	0	0
2025-04	senior	book	3	14.6667	18	2

4 rows in set (0.001 sec)

<u>Client Borrowing Report:</u>

Using 25 as a sample member ID.

SQL command:

```
select * from loan where memberID = 25;
select * from fine where memberID=25 and status="upaid";
select * from reservation where memberID=25 and status="active";
```

Output:

loanID	memberID	itemID	checkoutDate	dueDate	returnDate	lateFeeCharge
51	25	66	2025-04-27	2025-05-11	NULL	0
52	25	33	2025-04-27	2025-05-11	NULL	0
53	25	50	2025-04-27	2025-05-11	NULL	0
54	25	22	2025-04-27	2025-05-11	NULL	0
55	25	31	2025-04-27	2025-05-11	NULL	0

5 rows in set (0.000 sec)

```
MariaDB [447s25_j776w781]> select * from fine where memberID=25 and status="upaid";
Empty set (0.000 sec)

MariaDB [447s25_j776w781]> select * from reservation where memberID=25 and status="active";
```

reservationID	memberID	itemID	reservationDate	expirationDate	status
21	25	3	2025-05-04	2025-05-07	active
22	25	23	2025-05-04	NULL	active

<u>Item Availability and History:</u>

For the second table, 2-2-2022 is used as a cutoff date, but any date could be used.

SQL Query:

```
select m.itemID, title, availabilityStatus, checkoutDate as "Last Borrow
Date" from media m left join loan l
 on m.itemID = l.itemID where l.checkoutDate = (select MAX(checkoutDate)
from loan l2 where l2.itemID=l.itemID) or l.checkoutDate is null order
by itemID;

select m.itemID, title, availabilityStatus, checkoutDate as "Last Borrow
Date" from media m left join loan l
```

```
 on m.itemID = l.itemID where l.checkoutDate = (select MAX(checkoutDate)
from loan l2 where l2.itemID=l.itemID) and l.checkoutDate <
Date("2022-02-02") or l.checkoutDate is null order by itemID;
```

Output:

itemID	title	availabilityStatus	Last Borrow Date
1	The Wealth of Nations	available	2025-03-01
2	The Art of War	unavailable	2025-04-27
3	Blood Meridian	available	2022-02-21
4	Crime and Punishment	available	2022-02-13
5	Pride and Prejudice	available	2022-01-11
6	Ender's Game	available	2022-01-08
7	War and Peace	available	2022-01-27
8	No Longer Human	available	2022-02-19
9	The Great Gatsby	available	NULL
10	1984	available	2022-01-08
11	The book of Five Rings	available	NULL
12	The Tell-Tale Heart	available	2025-04-19
13	The Hobbit	available	2025-04-12
14	A Clockwork Orange	available	2022-01-22
15	Animal Farm	available	2022-02-12
16	The Road	available	2022-03-09
17	Heart of Darkness	available	2022-01-28
18	The Raven and Other Poems	available	NULL
19	The Odyssey	available	NULL
20	The Iliad	available	NULL
21	National Geographic	available	NULL
22	The New Yorker	unavailable	2025-04-27
23	Time	unavailable	2025-04-20
24	Scientific American	available	NULL
25	Forbes	available	NULL
26	The Economist	available	NULL
27	Wired	available	NULL
28	Harvard Business Review	available	NULL
29	Smithsonian	available	NULL
30	Vogue	available	NULL
31	The Atlantic	unavailable	2025-04-27
32	New Scientist	available	NULL
33	Popular Science	unavailable	2025-04-27
34	National Review	available	NULL
35	Discover	available	NULL
36	GQ	available	NULL
37	The Nation	available	NULL
38	PC Gamer	available	2022-01-05
39	Bon Appétit	available	NULL
40	Entertainment Weekly	available	NULL
41	Dune: Part Two	available	2022-02-08
42	Inside Out 2	available	NULL
43	Psycho	available	2022-03-25
44	The Godfather Part II	available	NULL
45	Forrest Gump	available	NULL
46	12 Angry Men	available	2022-02-02
47	2001: A Space Odyssey	available	NULL
48	Titanic	available	NULL
49	Saving Private Ryan	available	NULL
50	Rocky	unavailable	2025-04-27
51	The Sopranos: Season 1	available	NULL
52	Breaking Bad: Season 5	available	NULL
53	Friends: Season 10	available	NULL
54	Seinfeld: Season 9	available	NULL
55	The Simpsons: Season 34	available	NULL
56	The Wire: Season 5	available	2022-01-27
57	Better Call Saul: Season 6	available	NULL
58	Ted Lasso: Season 3	available	NULL
59	Only Murders in the Building: Season 3	available	2022-02-21
60	Buffy the Vampire Slayer: Season 7	available	NULL
61	The Hobbit	available	NULL
62	The Hobbit	available	NULL
63	The Road	available	NULL
64	The Hobbit	available	NULL
65	12 Angry Men	available	NULL
66	Necronomicon	unavailable	2025-04-27
67	Grays Sports Almanac	available	NULL

itemID	title	availabilityStatus	Last Borrow Date
5	Pride and Prejudice	available	2022-01-11
6	Ender's Game	available	2022-01-08
7	War and Peace	available	2022-01-27
9	The Great Gatsby	available	NULL
10	1984	available	2022-01-08
11	The book of Five Rings	available	NULL
14	A Clockwork Orange	available	2022-01-22
17	Heart of Darkness	available	2022-01-28
18	The Raven and Other Poems	available	NULL
19	The Odyssey	available	NULL
20	The Iliad	available	NULL
21	National Geographic	available	NULL
24	Scientific American	available	NULL
25	Forbes	available	NULL
26	The Economist	available	NULL
27	Wired	available	NULL
28	Harvard Business Review	available	NULL
29	Smithsonian	available	NULL
30	Vogue	available	NULL
32	New Scientist	available	NULL
34	National Review	available	NULL
35	Discover	available	NULL
36	GQ	available	NULL
37	The Nation	available	NULL
38	PC Gamer	available	2022-01-05
39	Bon Appétit	available	NULL
40	Entertainment Weekly	available	NULL
42	Inside Out 2	available	NULL
44	The Godfather Part II	available	NULL
45	Forrest Gump	available	NULL
47	2001: A Space Odyssey	available	NULL
48	Titanic	available	NULL
49	Saving Private Ryan	available	NULL
51	The Sopranos: Season 1	available	NULL
52	Breaking Bad: Season 5	available	NULL
53	Friends: Season 10	available	NULL
54	Seinfeld: Season 9	available	NULL
55	The Simpsons: Season 34	available	NULL
56	The Wire: Season 5	available	2022-01-27
57	Better Call Saul: Season 6	available	NULL
58	Ted Lasso: Season 3	available	NULL
60	Buffy the Vampire Slayer: Season 7	available	NULL
61	The Hobbit	available	NULL
62	The Hobbit	available	NULL
63	The Road	available	NULL
64	The Hobbit	available	NULL
65	12 Angry Men	available	NULL
67	Grays Sports Almanac	available	NULL

48 rows in set (0.000 sec)

Overdue Items Report:

Using May 12th, 2025, as the sample date, since all active loans would be overdue at this point.

SQL Query:

```
select l.loanID, i.itemID, i.title, l.checkoutDate, l.dueDate, u.userID,
u.name, (DATEDIFF(DATE("2025-05-12"), l.dueDate) * (i.specialPremium +
m.lateFeeRate)) as "Calculated Fine"  from ((user u nat
ural join member m) join loan l on u.userID=l.memberID) join media i on
l.itemID = i.itemID where l.returnDate is
 NULL and l.dueDate < DATE("2025-05-12");
```

Output:

loanID	itemID	title	checkoutDate	dueDate	userID	name	Calculated Fine
51	66	Necronomicon	2025-04-27	2025-05-11	25	LeBron James	11
52	33	Popular Science	2025-04-27	2025-05-11	25	LeBron James	4
53	50	Rocky	2025-04-27	2025-05-11	25	LeBron James	4
54	22	The New Yorker	2025-04-27	2025-05-11	25	LeBron James	4
55	31	The Atlantic	2025-04-27	2025-05-11	25	LeBron James	4
56	2	The Art of War	2025-04-27	2025-05-11	23	Puke Maresh	2
60	23	Time	2025-04-20	2025-05-04	7	Nate Ahmed	32

7 rows in set (0.001 sec)

Revenue Summary:

SQL Queries:

```
select m.typeName as "Membership Type", sum(p.amount) as "Total Revenue"
from payment p join member m on p.memberId = m.userID group by
m.typeName;

select m.itemType as "Media Type", sum(p.amount) as "Total Revenue" from
((media m join loan l on m.itemID=l.itemID) join fine f on f.loanID =
l.loanID) join payment p on p.fineID = f.fineID group by m.itemType;
```

Output:

```
ct m.ite+----------------+--------------+
| Membership Type | Total Revenue |
+-----------------+--------------+
| regular         |           29 |
| student         |            1 |
| senior          |           35 |
+-----------------+--------------+
3 rows in set (0.001 sec)
```

```
+-------------+----------------+
| Media Type | Total Revenue |
+-------------+----------------+
| book        |             60 |
| digital     |              5 |
+-------------+----------------+
2 rows in set (0.000 sec)
```

Problem Member Analysis:

Members with overdue loans

```
SELECT memberID, COUNT(*) AS overdueCount
FROM loan
WHERE returnDate > dueDate
GROUP BY memberID;
```

```
+----------+--------------+
| memberID | overdueCount |
+----------+--------------+
|        5 |            4 |
|        8 |            1 |
|       14 |            1 |
|       16 |            1 |
|       27 |            1 |
|       30 |            1 |
|       31 |            1 |
|       35 |            1 |
|       40 |            1 |
|       43 |            1 |
+----------+--------------+
10 rows in set (0.000 sec)
```

Members with unpaid/late fines

```
SELECT f.memberID, COUNT(*) AS unpaidFines
FROM fine f
LEFT JOIN payment p on f.fineID = p.fineID
WHERE f.status = 'unpaid'
GROUP BY f.memberID;
```

```
+----------+-------------+
| memberID | unpaidFines |
+----------+-------------+
|        5 |           2 |
|       27 |           1 |
|       43 |           1 |
+----------+-------------+
3 rows in set (0.000 sec)
```

Members who are currently borrowing items
(used so that the system doesn't recommend deactivating the accounts of users who currently possess library items)

```
SELECT unique memberID
FROM loan
WHERE returnDate IS NULL;
```

```
+----------+
| memberID |
+----------+
|        7 |
|       23 |
|       25 |
+----------+
3 rows in set (0.000 sec)
```

Collection Analysis:

Tends in acquisitions:

```
SELECT m.itemID, m.title, COUNT(l.itemID) AS loan_count
      FROM media m
      LEFT JOIN loan l ON m.itemID = l.itemID
      GROUP BY m.itemID, m.title
      ORDER BY loan_count DESC;
```

itemID	title	loan_count
43	Psycho	5
2	The Art of War	4
66	Necronomicon	4
16	The Road	4
13	The Hobbit	4
8	No Longer Human	3
4	Crime and Punishment	3
59	Only Murders in the Building: Season 3	3
15	Animal Farm	3
41	Dune: Part Two	3
3	Blood Meridian	3
17	Heart of Darkness	2
14	A Clockwork Orange	2
46	12 Angry Men	2
7	War and Peace	2
56	The Wire: Season 5	2
23	Time	1
5	Pride and Prejudice	1
31	The Atlantic	1
22	The New Yorker	1
10	1984	1
1	The Wealth of Nations	1
33	Popular Science	1
50	Rocky	1
12	The Tell-Tale Heart	1
6	Ender's Game	1
38	PC Gamer	1
26	The Economist	0
58	Ted Lasso: Season 3	0
55	The Simpsons: Season 34	0
20	The Iliad	0
52	Breaking Bad: Season 5	0
49	Saving Private Ryan	0
11	The book of Five Rings	0
40	Entertainment Weekly	0
37	The Nation	0
34	National Review	0
63	The Road	0
28	Harvard Business Review	0
60	Buffy the Vampire Slayer: Season 7	0
25	Forbes	0
57	Better Call Saul: Season 6	0
54	Seinfeld: Season 9	0
19	The Odyssey	0
51	The Sopranos: Season 1	0
48	Titanic	0
45	Forrest Gump	0
42	Inside Out 2	0

```
|     39 | Bon Appétit                          |           0 |
|     36 | GQ                                   |           0 |
|     65 | 12 Angry Men                         |           0 |
|     30 | Vogue                                |           0 |
|     62 | The Hobbit                           |           0 |
|     27 | Wired                                |           0 |
|     24 | Scientific American                  |           0 |
|     21 | National Geographic                  |           0 |
|     53 | Friends: Season 10                   |           0 |
|     18 | The Raven and Other Poems            |           0 |
|     47 | 2001: A Space Odyssey                |           0 |
|     44 | The Godfather Part II                |           0 |
|      9 | The Great Gatsby                     |           0 |
|     35 | Discover                             |           0 |
|     67 | Grays Sports Almanac                 |           0 |
|     32 | New Scientist                        |           0 |
|     64 | The Hobbit                           |           0 |
|     29 | Smithsonian                          |           0 |
|     61 | The Hobbit                           |           0 |
+--------+--------------------------------------+-------------+
```

Under represented Genre :

```
SELECT b.genre, COUNT(*) AS unloaned_count
FROM media m
     JOIN book b ON m.itemID = b.itemID
     LEFT JOIN loan l ON m.itemID = l.itemID
     WHERE l.itemID IS NULL
     GROUP BY b.genre
     ORDER BY unloaned_count DESC
     LIMIT 1;
```

```
+----------+----------------+
| genre    | unloaned_count |
+----------+----------------+
|  Fantasy |              3 |
+----------+----------------+
1 row in set (0.001 sec)
```

Under represented Item Type :

```
SELECT m.itemType, COUNT(*) AS unloaned_count
   FROM media m
   LEFT JOIN loan l ON m.itemID = l.itemID
   WHERE l.itemID IS NULL
   GROUP BY m.itemType
   ORDER BY unloaned_count DESC
   LIMIT 1;
```

```
+----------+----------------+
| itemType | unloaned_count |
+----------+----------------+
| magazine |             15 |
+----------+----------------+
1 row in set (0.000 sec)
```