Library Management: SQL Analysis

Query 1: Top 5 Most Borrowed Books
Business Question: Which books are the most popular among library members?
Insight: Identifies high-demand books that may require additional copies or help inform future purchasing decisions.

Code:

```
Select b.title, b.author,
COUNT(c.checkout_ID) as times_borrowed
from books b
LEFT join checkouts c on c.book_id = b.book_id
GROUP BY b.book_id, b.title, b.author
ORDER BY times_borrowed DESC
LIMIT 5;
```

Output:

title	author	times_borrowed
To Kill a Mockingbird	Harper Lee	2
1984	George Orwell	2
Harry Potter and the Sorcerer's Stone	J.K. Rowling	2
The Hobbit	J.R.R. Tolkien	2
The Catcher in the Rye	J.D. Salinger	2

Query 2: Members with Overdue Books
Business Question: Which members currently have overdue books?
Insight: Enables the library to send targeted reminders and follow up with members who have outstanding items, improving return rates.

Code:

```
Select m.member_id, m.first_name, m.last_name, c.status
from members m
left JOIN checkouts c on m.member_id = c.member_id
where c.status = 'Overdue';
```

Output:

member_id	first_name	last_name	status
3	Michael	Brown	Overdue
5	James	Wilson	Overdue
13	Amanda	Lopez	Overdue
15	Michelle	Walker	Overdue
21	Nicole	Scott	Overdue
25	Rebecca	Nelson	Overdue

Query 3: Count of Borrowed vs. Returned Books

Business Question: What is the current status of all checkouts (borrowed vs. returned)?

Insight: Provides a snapshot of library circulation activity and helps track inventory availability.

Code:

Select count(b.book_id), c.status

from books b

join checkouts c on b.book_id = c.book_id

where c.status = 'Borrowed' or c.status = 'Returned'

GROUP by c.status;

Output:

count(b.book_id)	status
14	Borrowed
10	Returned

Query 4: Top 5 Most Popular Genres

Business Question: Which genres are most frequently borrowed by members?

Insight: Helps inform collection development strategy by identifying genre preferences and potential gaps in the catalog.

Code:

Select b.genre, COUNT(c.checkout_ID) as times_borrowed

from books b

join checkouts c on b.book_id = c.book_id

GROUP by b.genre

ORDER by times_borrowed DESC

limit 5;

Output:

genre	times_borrowed
Fantasy	7
Dystopian	5
Fiction	4
Romance	2
Political Satire	2

Query 5: Active Members with No Borrowing Activity

Business Question: Which active members have never borrowed a book?

Insight: Identifies potentially disengaged members who may benefit from outreach or personalized recommendations to encourage library usage.

Code:

```
Select m.member_id, m.first_name, m.last_name, m.membership_status
from members m
left join checkouts c on m.member_id = c.member_id
where m.membership_status = 'Active' AND c.checkout_id IS NULL;
```

Output:

member_id	first_name	last_name	membership_status
10	Mary	White	Active

Query 6: Members Who Borrowed More Than 1 Book

Business Question: Which members are frequent borrowers (more than 1 book)?

Insight: Identifies engaged library users who could be candidates for reading programs, loyalty initiatives, or feedback opportunities.

Code:

```
Select COUNT(c.checkout_id) as books_borrowed, m.first_name, m.last_name
from checkouts c
left join members m on c.member_id = m.member_id
GROUP by c.member_id
having books_borrowed >1;
```

Output:

books_borrowed	first_name	last_name
2	Emma	Johnson
2	Patricia	Garcia
2	Christopher	Rodriguez
2	Michelle	Walker
2	Brian	Young
2	Jason	Wright

Query 7: Average Borrowing Duration

Business Question: What is the average number of days members keep books before returning them?

Insight: Helps optimize loan periods and understand typical borrowing behavior patterns.

Technical Note: Initially, direct date subtraction returned incorrect results (0.1) because dates were stored as TEXT. Used julianday() function to properly calculate the difference in days between checkout and return dates.

Code:

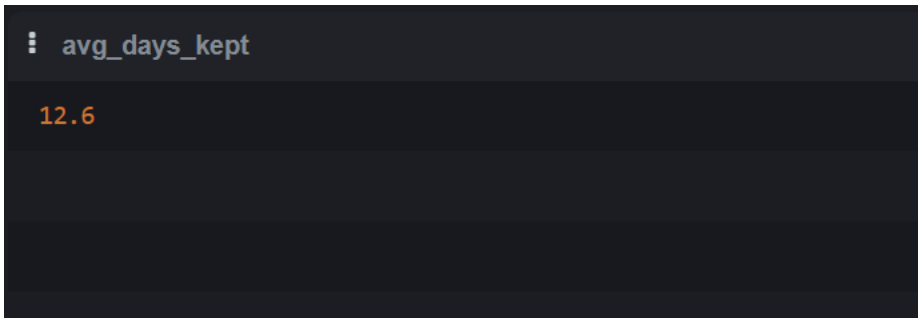
```
Select AVG(julianday(return_date) - julianday(checkout_date)) AS avg_days_kept

FROM checkouts

WHERE return_date IS NOT NULL;
```

Output:

Note: Initially, direct date subtraction returned incorrect results (0.1) because dates were stored as TEXT. Used julianday() function to properly calculate the difference in days between checkout and return dates.



Query 8: Overdue Books with Member Contact Information

Business Question: Which books are currently overdue, and how can we contact the members?

Insight: Provides actionable data for recovery efforts, including member contact details for follow-up communications.

Code:

Select m.member_id,m.first_name, m.last_name, m.phone, m.email, c.status, b.title

from members m

Left JOIN checkouts c on m.member_id = c.member_id

left JOIN books b ON c.book_id = b.book_id

where c.status = 'Overdue'

order by m.member_id;

Output:

member_id	first_name	last_name	phone	email	status	title
3	Michael	Brown	555-0103	mbrown@email.com	Overdue	The Great Gatsby
5	James	Wilson	555-0105	jwilson@email.com	Overdue	Pride and P rejudice
13	Amanda	Lopez	555-0113	alopez@email.com	Overdue	The Odyssey
15	Michelle	Walker	555-0115	mwalker@email.com	Overdue	Crime and Punishment
21	Nicole	Scott	555-0121	nscott@email.com	Overdue	The Picture of Dorian Gray
25	Rebecca	Nelson	555-0125	rnelson@email.com	Overdue	Fahrenheit 451

Query 9: Total Late Fee Revenue

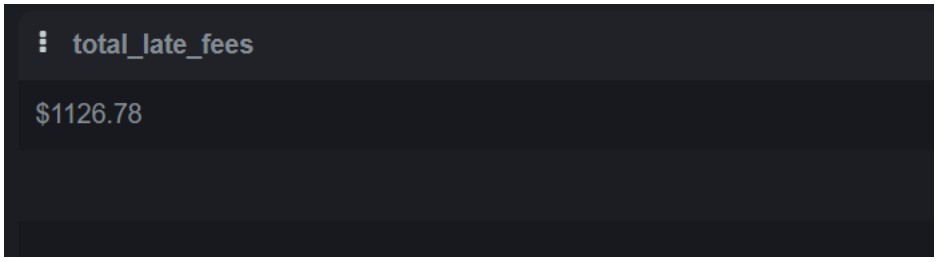
Business Question: What is the total amount of late fees owed or collected (at $0.50 per day)?

Insight: Quantifies potential revenue from late returns and helps track financial impact of overdue items.

Code:

SELECT '$' || ROUND(SUM (CASE WHEN c.status = 'Returned' THEN 0.5*(julianday(return_date)-julianday(due_date)) WHEN c.status = 'Overdue' THEN 0.5*(julianday('now')-julianday(due_date)) ELSE 0 END) , 2) AS total_late_fees FROM checkouts;

Output:



Query 10: Authors with Highest Return Rate

Business Question: Which authors have the highest percentage of books returned vs. still borrowed?

Insight: Identifies authors whose books circulate efficiently, potentially indicating strong reader satisfaction and completion rates.

Code:

```
SELECT
   b.author,
   COUNT(c.checkout_id) AS total_checkouts,
   COUNT(CASE WHEN c.status = 'Returned' THEN 1 END) AS returned_count,
   ROUND(
   (COUNT(CASE WHEN c.status = 'Returned' THEN 1 END) * 100.0 /
COUNT(c.checkout_id)), 2) AS return_rate
FROM books b
JOIN checkouts c ON b.book_id = c.book_id
GROUP BY b.author
HAVING returned_count > 0
ORDER BY return_rate DESC
LIMIT 10;
```

Output:

author	total_checkouts	returned_count	return_rate
Lewis Carroll	1	1	100
Emily Brontë	1	1	100
Aldous Huxley	1	1	100
J.R.R. Tolkien	3	2	66.67
J.K. Rowling	2	1	50
J.D. Salinger	2	1	50
Harper Lee	2	1	50
George Orwell	4	2	50

Query 11: Member Rankings by Borrowing Activity

Business Question: How do members rank based on total books borrowed?

Insight: Recognizes the most active library users and helps segment members by engagement level.

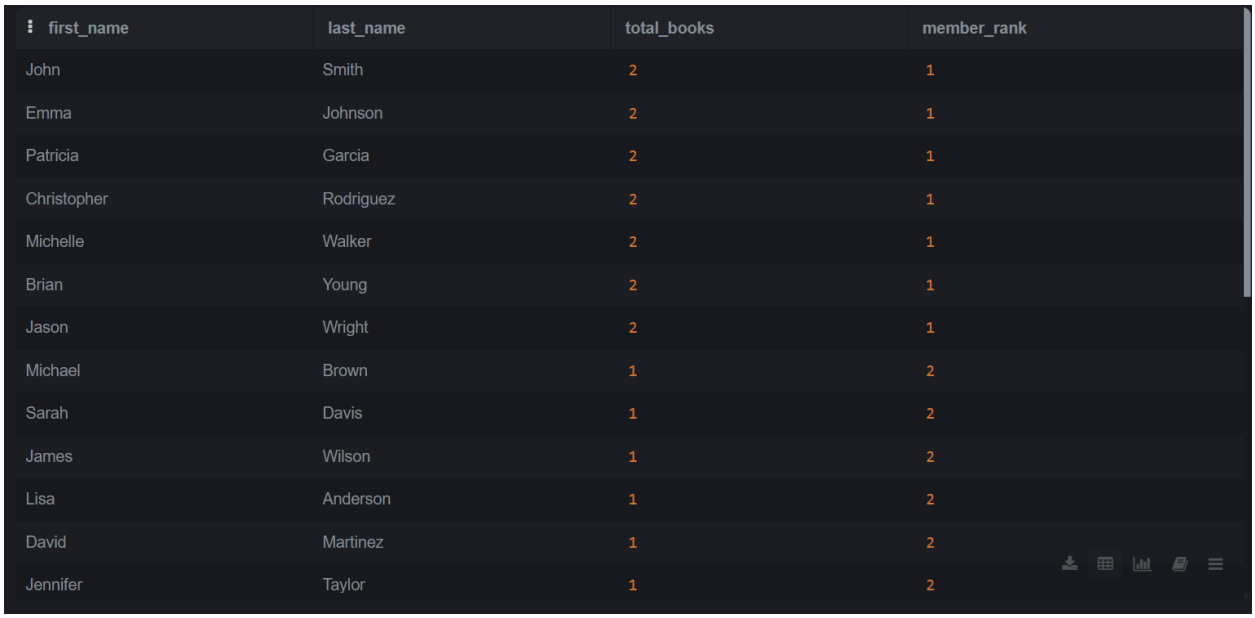
Technical Note: Initially used RANK() function which resulted in rank gaps after ties (e.g., multiple members ranked 1, next member ranked 8). Switched to DENSE_RANK() to ensure consecutive ranking without gaps (1, 1, 1, 2, 2...), providing more intuitive results when multiple members have the same borrowing count.

Code:

```
SELECT
  m.first_name, m.last_name,
  COUNT(c.checkout_id) as total_books,
  Dense_RANK() OVER (ORDER BY COUNT(c.checkout_id) DESC) as member_rank
FROM checkouts c
join members m on m.member_id = c.member_id
```

GROUP BY m.member_id, m.first_name, m.last_name;

Output:



first_name	last_name	total_books	member_rank
John	Smith	2	1
Emma	Johnson	2	1
Patricia	Garcia	2	1
Christopher	Rodriguez	2	1
Michelle	Walker	2	1
Brian	Young	2	1
Jason	Wright	2	1
Michael	Brown	1	2
Sarah	Davis	1	2
James	Wilson	1	2
Lisa	Anderson	1	2
David	Martinez	1	2
Jennifer	Taylor	1	2

Query 12: Running Total of Checkouts by Book

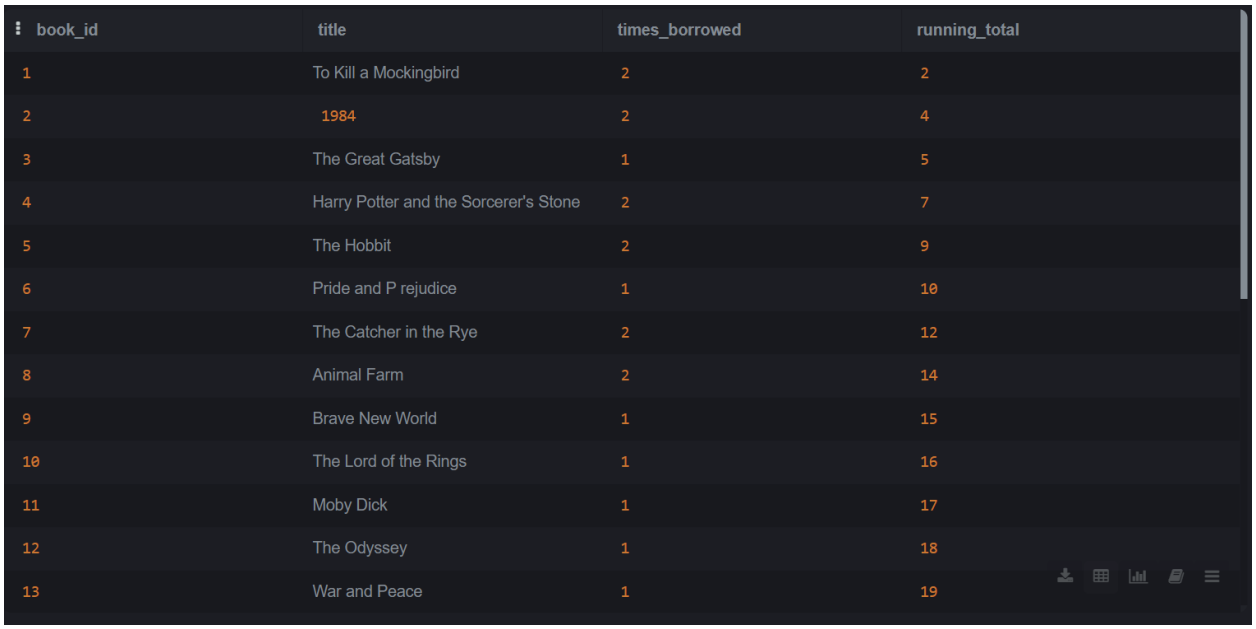
Business Question: What is the cumulative checkout count across all books?

Insight: Demonstrates circulation trends over the entire collection using window functions to show running totals.

Code:

```
SELECT b.book_id, b.title,
    COUNT(c.checkout_id) AS times_borrowed,
    SUM(COUNT(c.checkout_id)) OVER (ORDER BY b.book_id) AS running_total
FROM books b
JOIN checkouts c on b.book_id = c.book_id
GROUP by b.book_id, b.title;
```

Output:



book_id	title	times_borrowed	running_total
1	To Kill a Mockingbird	2	2
2	1984	2	4
3	The Great Gatsby	1	5
4	Harry Potter and the Sorcerer's Stone	2	7
5	The Hobbit	2	9
6	Pride and P rejudice	1	10
7	The Catcher in the Rye	2	12
8	Animal Farm	2	14
9	Brave New World	1	15
10	The Lord of the Rings	1	16
11	Moby Dick	1	17
12	The Odyssey	1	18
13	War and Peace	1	19

Query 13: Books Borrowed Above Average
Business Question: Which books have been borrowed more frequently than the average?
Insight: Highlights exceptional performers in the collection that exceed typical circulation rates, indicating high demand titles.

Code:

```
SELECT
  b.book_id,
  b.title,
  COUNT(c.checkout_id) AS times_borrowed
FROM books b
JOIN checkouts c ON b.book_id = c.book_id
```

```
GROUP BY b.book_id, b.title
HAVING COUNT(c.checkout_id) > (
  SELECT AVG(checkout_count)
  FROM (
    SELECT COUNT(checkout_id) AS checkout_count
    FROM checkouts
    GROUP BY book_id));
```

Output:

book_id	title	times_borrowed
1	To Kill a Mockingbird	2
2	1984	2
4	Harry Potter and the Sorcerer's Stone	2
5	The Hobbit	2
7	The Catcher in the Rye	2
8	Animal Farm	2