Here are some advanced queries for the given database structure:

1. List customers who have overdue books:

```sql

SELECT c.first\_name, c.last\_name, b.title, i.due\_date

FROM customers c

JOIN issued\_books i ON c.id = i.customer\_id

JOIN books b ON i.book\_id = b.id

WHERE i.return\_date IS NULL AND i.due\_date < CURRENT\_DATE;

```

2. List the most popular books (most issued):

```sql

SELECT b.title, COUNT(i.book\_id) AS total\_issued

FROM books b

JOIN issued\_books i ON b.id = i.book\_id

GROUP BY b.title

ORDER BY total\_issued DESC;

```

3. List customers who have never issued a book:

```sql

SELECT \* FROM customers

WHERE id NOT IN (SELECT customer\_id FROM issued\_books);

```

4. Update book price by 10% for books published before 2000:

```sql

UPDATE books

SET price = price \* 1.1

WHERE title IN (SELECT title FROM books WHERE EXTRACT(YEAR FROM publication\_date) < 2000);

```

5. Delete all overdue issued book records:

```sql

DELETE FROM issued\_books

WHERE due\_date < CURRENT\_DATE AND return\_date IS NULL;

```

6. List total revenue generated from issued books:

```sql

SELECT SUM(b.price) AS total\_revenue

FROM issued\_books i

JOIN books b ON i.book\_id = b.id;

```

7. List the number of copies left for each book:

```sql

SELECT b.title, (b.total\_copies - COUNT(i.book\_id)) AS copies\_left

FROM books b

LEFT JOIN issued\_books i ON b.id = i.book\_id

GROUP BY b.title;

```

These are some advanced queries that can help you build functionality for the Library Management System. Let me know if you have any other questions!

Certainly! Here are some advanced examples of queries you can run on the "books," "customers," and "issued\_books" tables:

1. Retrieve all books with their authors and categories:

```sql

SELECT title, author, category

FROM books;

```

2. Find books with a price higher than $50:

```sql

SELECT title, price

FROM books

WHERE price > 50.00;

```

3. Get the total number of books in each category:

```sql

SELECT category, COUNT(\*) AS total\_books

FROM books

GROUP BY category;

```

4. Find customers with the last name "Smith":

```sql

SELECT first\_name, last\_name, email

FROM customers

WHERE last\_name = 'Smith';

```

5. Retrieve issued books along with their due dates for a specific customer:

```sql

SELECT b.title, ib.due\_date

FROM issued\_books ib

JOIN books b ON b.id = ib.book\_id

WHERE ib.customer\_id = 1;

```

6. Calculate the average price of books in each category:

```sql

SELECT category, AVG(price) AS avg\_price

FROM books

GROUP BY category;

```

7. Update the status of a book with a specific ID:

```sql

UPDATE books

SET status = 'Unavailable'

WHERE id = 1;

```

8. Delete an issued book record with a specific ID:

```sql

DELETE FROM issued\_books

WHERE id = 1;

```

These examples demonstrate various types of queries you can run on the given tables, including selecting data, filtering based on conditions, joining tables, aggregating data, updating records, and deleting records. Feel free to modify them according to your specific requirements or expand upon them to suit your needs.

Certainly! Here are a few more query examples for the "books," "customers," and "issued\_books" tables:

1. Retrieve all customers along with the number of books they have issued:

```sql

SELECT c.first\_name, c.last\_name, COUNT(ib.book\_id) AS total\_books\_issued

FROM customers c

LEFT JOIN issued\_books ib ON ib.customer\_id = c.id

GROUP BY c.id;

```

2. Find books with the highest price in each category:

```sql

SELECT category, MAX(price) AS max\_price

FROM books

GROUP BY category;

```

3. Retrieve the details of books that are currently issued and their corresponding customers:

```sql

SELECT b.title, c.first\_name, c.last\_name

FROM books b

JOIN issued\_books ib ON ib.book\_id = b.id

JOIN customers c ON c.id = ib.customer\_id;

```

4. Calculate the total revenue generated from book sales:

```sql

SELECT SUM(price) AS total\_revenue

FROM books;

```

5. Find customers who have not issued any books:

```sql

SELECT c.first\_name, c.last\_name

FROM customers c

LEFT JOIN issued\_books ib ON ib.customer\_id = c.id

WHERE ib.customer\_id IS NULL;

```

6. Retrieve books that have not been issued yet:

```sql

SELECT title, author, category

FROM books

LEFT JOIN issued\_books ib ON ib.book\_id = books.id

WHERE ib.book\_id IS NULL;

```

These additional query examples cover a range of scenarios, such as calculating aggregates, joining multiple tables, identifying missing data, and retrieving specific subsets of information. Feel free to adapt and modify them as needed to suit your specific use cases.