SQL queries (10 points each)

1. **Fine Calculation:** Calculate the total fines owed by each member, considering overdue books and a daily fine rate (e.g., $0.25 per day).

```
MariaDB [447f24_m913m292]> select m.Member_ID, m.Member_name_first, m.Member_name_last, sum(DATEDIFF(CURDATE(), co.
Due_date) * 0.25) as Total_Fines from Checks_Out co JOIN Member m on co.Member_ID = m.Member_ID where co.Due_date <
 CURDATE() group by m.Member_ID, m.Member_name_first, m.Member_name_last;
+-----------+------------------+-----------------+-------------+
| Member_ID | Member_name_first | Member_name_last | Total_Fines |
+-----------+------------------+-----------------+-------------+
|      1462 | Isabella          | Walker          |        3.75 |
|      2765 | Ava               | Bennett         |        6.75 |
|      3912 | Noah              | Thompson        |        0.75 |
|      5837 | Mason             | Miller          |        3.25 |
|      6521 | Sophia            | Reed            |      138.50 |
|      7489 | Emma              | Harrison        |        7.00 |
|      8341 | Liam              | Davis           |        2.75 |
+-----------+------------------+-----------------+-------------+
7 rows in set (0.001 sec)
```

The query input here was chosen to display the member's ID, first name, last name, and the total fines which were calculated adding up the fines already there and also applying a 25 cents late fine. It checked who had a book checked out where the due date had already passed. In doing so, it found this list of people and calculated their total fines based on how late they are.

2. **Book Availability:** Display a list of all available books (not currently borrowed) within a specific genre.

```
MariaDB [447f24_m913m292]> select m.Title, m.Genre from Medium m left join Checks_Out co on m.ID = co.Medium_ID whe
re co.Medium_ID is null and m.Genre = 'Historical Fiction';
+--------------------+--------------------+
| Title              | Genre              |
+--------------------+--------------------+
| Les Misérables     | Historical Fiction |
| The Grapes of Wrath | Historical Fiction |
| Anna Karenina      | Historical Fiction |
| The Scarlet Letter | Historical Fiction |
+--------------------+--------------------+
4 rows in set (0.001 sec)
```

This query displays the title and genre of the books we chose to look for. In this case, we chose to find the Historical Fiction books which were not checked out yet. The query found this by searching the Medium and Checks_Out tables for any books with a NULL ID (which implies no one associated with the book).

3. **Frequent Borrowers of a Specific Genre:** Identify the members who have borrowed the most books in a particular genre (e.g., "Mystery") in the last year.

```
MariaDB [447f24_m913m292]> SELECT co.Member_ID, COUNT(*) AS Borrowed_Books FROM Checks_Out co JOIN Medium m ON co.M
edium_ID = m.ID WHERE m.Genre = 'Dystopian' AND co.Checkout_date > CURDATE() - INTERVAL 1 YEAR GROUP BY co.Member_I
D ORDER BY Borrowed_Books DESC LIMIT 1;
+-----------+----------------+
| Member_ID | Borrowed_Books |
+-----------+----------------+
|      4598 |              1 |
+-----------+----------------+
1 row in set (0.001 sec)
```

This query displays the Member_ID and borrowed books from that member. It also chooses a specific genre which is Dystopian in this case. The displayed results show the members who borrowed the most Dystopian books in the last year. The last year was calculated by setting the interval to 1 year. Dystopian books were only borrowed once throughout the list of IDs and it was from Member_ID 4598.

4. **Books Due Soon:** Generate a report of all books due within the next week, sorted by due date.

```
MariaDB [447f24_m913m292]> select m.Title, co.Due_date from Checks_Out co join Medium m on co.Medium_ID = m.ID wher
e co.Due_date between CURDATE() and CURDATE() + INTERVAL 7 DAY order by co.Due_date;
+---------------------+------------+
| Title               | Due_date   |
+---------------------+------------+
| 1984                | 2024-12-14 |
| Pride and Prejudice | 2024-12-15 |
| Crime and Punishment | 2024-12-15 |
+---------------------+------------+
3 rows in set (0.001 sec)
```

The query displays the title and due date of books that are due within the next week. It did this by joining the Checks_Out and Medium tables at the IDs and finding the current date + 7 days in order of Due_date.

5. **Members with Overdue Books:** List all members who currently have at least one overdue book, along with the titles of the overdue books.

```
MariaDB [447f24_m913m292]> select m.Member_name_first, m.Member_name_last, mb.Title from Member m join Checks_Out c
o on m.Member_ID = co.Member_ID join Medium mb on co.Medium_ID = mb.ID where co.Due_date < CURDATE() order by m.Mem
ber_name_last;
+-------------------+------------------+-----------------------+
| Member_name_first | Member_name_last | Title                 |
+-------------------+------------------+-----------------------+
| Ava               | Bennett          | Moby-Dick             |
| Liam              | Davis            | To Kill a Mockingbird |
| Emma              | Harrison         | War and Peace         |
| Mason             | Miller           | The Catcher in the Rye |
| Sophia            | Reed             | The Lord of the Rings |
| Noah              | Thompson         | The Great Gatsby      |
| Isabella          | Walker           | Jane Eyre             |
+-------------------+------------------+-----------------------+
7 rows in set (0.002 sec)
```

This query printed the member first names, last names, and titles of the books that are overdue. These names seen above match the names in query #1 where the total fines were calculated because this is a similar query except we're just listing the names of the overdue books these people own. It did this by joining Member and Checks_Out and matching the IDs to each other. Then, it checked the due date to see if it was overdue. If the due date came up as being less than the current date, it flagged it as overdue and added the book title to the list along with the name of the person.

6. **Average Borrowing Time:** Calculate the average number of days members borrow books for a specific genre.

```
MariaDB [447f24_m913m292]> select avg(DATEDIFF(co.Due_date, co.Checkout_date)) as Average_Borrowing_Time from Check
s_Out co join Medium m on co.Medium_ID = m.ID where m.Genre = 'Crime';
+-----------------------+
| Average_Borrowing_Time |
+-----------------------+
|               14.0000 |
+-----------------------+
1 row in set (0.002 sec)
```

The average borrowing time was calculated by taking the average of the due dates compared to the checkout dates. The crime genre was chosen in this case.

7. **Most Popular Author in the Last Month:** Determine the author whose books have been borrowed the most in the last month.

```
MariaDB [447f24_m913m292]> select p.Author_name_first, p.Author_name_last, count(*) as Books_Borrowed from Checks_O
ut co join Medium m on co.Medium_ID = m.ID join Writes w on m.ID = w.Book_ID join Person p on w.Author_ID = p.Autho
r_ID where co.Checkout_date > CURDATE() - INTERVAL 1 MONTH group by p.Author_ID order by Books_Borrowed DESC LIMIT
1;
+------------------+-----------------+---------------+
| Author_name_first | Author_name_last | Books_Borrowed |
+------------------+-----------------+---------------+
| George           | Orwell          |             1 |
+------------------+-----------------+---------------+
1 row in set (0.002 sec)
```

This query displays the author's first and last name along with how many books of theirs were borrowed. Only one author is displayed and the Books_Borrowed is 1 because there are not multiple books checked out for one author. Table displayed is accurate. If multiple members owned George Orwell's books, that number under "Books_Borrowed" would change to 2.