Q1 Writing Queries
65 Points

The following questions will all use the schema for a library database as follows.

Books (isbn, title, author, genre, publisher)
Members (id, name)
Lending (isbn, id, checkout, returned) // checkout and returned are both DATETIME

A tuple in Books represents a single book in the library system (We're assuming the library doesn't have multiple copies of a book, and each book has a single author and publisher). There may be multiple books with the same title, and different isbn.

The Members table holds identifying information for people checking out books from the library. There may be multiple people with the same name, and different member ids.

Lastly the Lending table shows a history of books loaned to various members. The checkout and returned attributes are the time stamps the book was checked out to a user and subsequently returned by that user. Any Lending tuple with a NULL "returned" attribute means that book is still currently checked out. People may return a book and then check out the same book out again, each unique checkout becomes a new tuple with the new checkout time.

Books.isbn is the primary key for **Books**.
Members.id is the primary key for **Members**.

Lending.isbn is a foreign key to **Books**.isbn
Lending.id is a foreign key to **Members**.id.
Lending.isbn, **Lending**.id, and **Lending**.checkout together constitute the primary key for **Lending**.

isbn and id are INT type. checkout and returned are DATETIME type. All other attributes are VARCHAR(1000)

For all questions you are allowed to use sub-queries, but they may not be required.

Q1.1
15 Points

Write the CREATE TABLE(...) statement for the **Lending** table in the schema described above. Use DATETIME for the type of the *checkout* and *returned* attributes, and INT for *isbn* and *id*. Remember to specify the relevant primary key and foreign key constraints. (You don't have to write .import or 'PRAGMA FOREIGN KEYS')

```
CREATE TABLE Lending(
isbn INT,
id INT,
checkout DATETIME,
returned DATETIME,
PRIMARY KEY(isbn,id,checkout)
FOREIGN KEY(isbn) REFERENCES Books(isbn)
FOREIGN KEY(id) REFERENCES Members(id)
);

CREATE TABLE Books(
isbn INT PRIMARY KEY,
title VARCHAR(1000),
author VARCHAR(1000),
genre VARCHAR(1000),
publisher VARCHAR(1000));

CREATE TABLE Members(
  id INT PRIMARY KEY,
  name VARCHAR(1000));
```

Q1.2
15 Points

Write a query to find the books **currently** checked out and the name of the person checking out each one. Return the member name and book title, both sorted alphabetically. (Hint: the 'IS NULL' boolean test is useful here)

```
SELECT M.name, B.title
FROM Books B, Lending L, Members M
WHERE M.id = L.id
AND B.isbn = L.isbn
AND L.returned IS NULL
ORDER BY M.name, B.title ASC;
```

Q1.3
15 Points

Write a query to find the number of book checkouts that have occurred in each genre of book in the library. (Multiple checkouts of the same book are still multiple checkouts in that genre.) If there is a genre that has some books in the library but none have ever been checked out, that genre's count should be 0 in the output. Return the genre and number of checkouts, sorted by number of checkouts in decreasing order.

```
SELECT DISTINCT(B.genre), COUNT(L.checkout) as number_checkouts
FROM  Books B LEFT OUTER JOIN Lending L
ON L.isbn = B.isbn
GROUP BY B.genre
ORDER BY number_checkouts DESC;
```

Q1.4
15 Points

Find the names and ids of all people who checked out the book titled 'Leaves of Grass' and also checked out the book titled 'Harmonium' at some point (it is not required that the person had both books at once.) Return the id and name for each person who has checked out both books. Do not return duplicate tuples.

```
SELECT DISTINCT M.id, M.name
FROM Members M, Lending L , Books B
WHERE B.title = 'Leaves of Grass'
AND B.title = 'Harmonium'
AND L.id = M.id
AND L.isbn = B.isbn;
```

Q1.5 Short Answer
5 Points

Explain in a sentence or two why relational algebra is a useful query language compared to SQL.

Relational Algebra is a way that is able to describe imperative/procedural programming on the tables without the physical details of the execution. Relational algebra is a useful query language that reads more like instructions but still captures the fundamental operations of a query and explains the how of the query id executed. Since the code has to boil down to instructions at some point during execution, relational algebra is useful compared to SQL which is declarative.

Q2 True/False

5 Points

Answer true or false for the following questions about SQL and the relational model

Q2.1

1 Point

Subqueries in the FROM clause can return multiple attributes.

- True
- False

Q2.2

1 Point

The relational model uses static types for the attributes in a table.

- True
- False

Q2.3

1 Point

In relational algebra, grouping and aggregation can be done with a single operator.

- True
- False

Q2.4

1 Point

Every relational algebra plan uses the "project" operator.

- True
- False

Q2.5
1 Point

Subqueries in the WHERE clause must only return one attribute.

True

False

Midterm Quiz

● Graded

Select each question to review feedback and grading details.

Student
Samartha Ramkumar

Total Points
61 / 70 pts

Question 1
Writing Queries — **57** / 65 pts

1.1 (no title) — **15** / 15 pts

1.2 (no title) — **15** / 15 pts

1.3 (no title) — **15** / 15 pts

1.4 **(no title)** — **7** / 15 pts

- ✔ **– 2 pts** Missing/Wrong JOIN condition
- ✔ **– 2 pts** Missing/Wrong WHERE condition
- ✔ **– 2 pts** Need 2 Books
- ✔ **– 2 pts** need 2 lending tables

1.5 Short Answer — **5** / 5 pts

Question 2

True/False **4** / 5 pts

2.1	(no title)	**1** / 1 pt
2.2	(no title)	**1** / 1 pt
2.3	(no title)	**1** / 1 pt
2.4	(no title)	**1** / 1 pt
2.5	(no title)	**0** / 1 pt