Turza Saha
1001934520

Ethan Debnath
1001955946

Databases Project 2 Part 2

HONOR CODE:
I pledge, in my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence. I promise that I will submit only work that I personally create or that I contribute to group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of integrity and uphold the spirit of the Honor Code.

Contribution List:

Turza:
Documentation
Create Tables and Database Setup
Question 5-10

Ethan:
Documentation
Question 1-4b
Assumptions and Challenges

Task 1:

THE CREATE TABLE STATEMENTS:

```
CREATE TABLE IF NOT EXISTS BOOK(
        Book_Id integer PRIMARY KEY,
   Title TEXT NOT NULL,
   Publisher_name TEXT NOT NULL,

   FOREIGN KEY(Publisher_name) REFERENCES PUBLISHER(Publisher_name)
);
```

Comments: Book_Id is the Primary Key, as it connects the book information on other tables. Many tables later will refer to the Book_ID such as BOOK_LOANS, BOOK_COPIES etc. Publisher_Name foreign key to connect with PUBLISHER table.
We assumed no column can have null values.

```
CREATE TABLE IF NOT EXISTS PUBLISHER(
        Publisher_name TEXT PRIMARY KEY,
   Phone TEXT NOT NULL,
   Address TEXT NOT NULL
);
```

Comments: This table has the Publisher information with Publisher_Name as Primary Key, referenced in Book_Id. Phone and Address are both TEXT type.
We assumed no column can have null values.

```
CREATE TABLE IF NOT EXISTS LIBRARY_BRANCH(
        Branch_Id integer PRIMARY KEY,
   Branch_Name TEXT NOT NULL,
   Branch_Address TEXT NOT NULL

);
```

Comments: Branch_Id is the Primary Key
Note: We did not use the AUTO_INCREMENT word because it was not working. Rather, just putting Branch_Id as type “integer” and as PRIMARY KEY auto incremented the value upon insertion of new branches as shown on Query 4-b.
We assumed no column can have null values.

```
CREATE TABLE IF NOT EXISTS BORROWER(
        Card_No integer PRIMARY KEY,
   Name TEXT NOT NULL,
   Address TEXT NOT NULL,
   Phone TEXT NOT NULL
);
```

Comments: Card_No is the Primary Key to uniquely identify each Borrower. We assumed no column can have null values. Phone and Address are TEXT values for simplicity.
Note: We did not use the AUTO_INCREMENT word because it was not working. Rather, just putting Card_No as type “integer” and as PRIMARY KEY auto incremented the value upon insertion of new borrower as shown on Question 1 result.

```
CREATE TABLE IF NOT EXISTS BOOK_AUTHORS(
```

```
        Book_id INT NOT NULL,
    Author_name TEXT NOT NULL,

    FOREIGN KEY (Book_id) REFERENCES BOOK(Book_Id)
);
```

Comments: Book_id has to be foreign key and Author_name cannot be primary as there can be multiple books with the same author. A table without a primary key. Book_id references Book_id in BOOK

```
CREATE TABLE IF NOT EXISTS BOOK_LOANS(
        Book_ID INT NOT NULL,
    Branch_Id INT NOT NULL,
    Card_No INT NOT NULL,
    Date_Out TEXT NOT NULL,
    Due_Date TEXT NOT NULL,
    Returned_date TEXT,

    FOREIGN KEY(Book_Id) REFERENCES BOOK(Book_Id),
    FOREIGN KEY(Branch_Id) REFERENCES LIBRARY_BRANCH(Branch_Id),
    FOREIGN KEY(Card_No) REFERENCES BORROWER(Card_No)
);
```

Comments: Another table with no Primary Key. This table has three foreign keys, essentially the most important table connecting BOOK, LIBRARY_BRANCH and BORROWER.
Also, unlike other attributes, Returned_date may and can have NULL values as some loaned books have not been returned.

```
CREATE TABLE IF NOT EXISTS BOOK_COPIES(
        Book_Id INT NOT NULL,
    Branch_Id INT NOT NULL,
    No_of_copies INT NOT NULL,

    FOREIGN KEY(Book_Id) REFERENCES BOOK(Book_Id),
    FOREIGN KEY(Branch_Id) REFERENCES LIBRARY_BRANCH(Branch_Id)
);
```

Comments: Another table with no Primary Key. Two foreign keys to connect with BOOK(Book_Id) and LIBRARY_BRANCH(Branch_Id). So each book copy can be traced to what book and at which library.

Schema picture result:

```
sqlite> .schema
CREATE TABLE BOOK(
        Book_Id integer PRIMARY KEY,
    Title TEXT NOT NULL,
    Publisher_name TEXT NOT NULL,
    FOREIGN KEY(Publisher_name) REFERENCES PUBLISHER(Publisher_name)
);
CREATE TABLE PUBLISHER(
        Publisher_name TEXT PRIMARY KEY,
    Phone TEXT NOT NULL,
    Address TEXT NOT NULL
);
CREATE TABLE LIBRARY_BRANCH(
        Branch_Id integer PRIMARY KEY,
    Branch_Name TEXT NOT NULL,
    Branch_Address TEXT NOT NULL

);
CREATE TABLE BORROWER(
        Card_No integer PRIMARY KEY,
    Name TEXT NOT NULL,
    Address TEXT NOT NULL,
    Phone TEXT NOT NULL
);
CREATE TABLE BOOK_AUTHORS(
        Book_id INT NOT NULL,
    Author_name TEXT NOT NULL,

    FOREIGN KEY (Book_id) REFERENCES BOOK(Book_Id)
);
CREATE TABLE BOOK_LOANS(
        Book_ID INT NOT NULL,
    Branch_Id INT NOT NULL,
    Card_No INT NOT NULL,
    Date_Out TEXT NOT NULL,
    Due_Date TEXT NOT NULL,
    Returned_date TEXT,

    FOREIGN KEY(Book_Id) REFERENCES BOOK(Book_Id),
    FOREIGN KEY(Branch_Id) REFERENCES LIBRARY_BRANCH(Branch_Id),
    FOREIGN KEY(Card_No) REFERENCES BORROWER(Card_No)
);
CREATE TABLE BOOK_COPIES(
        Book_Id INT NOT NULL,
    Branch_Id INT NOT NULL,
    No_of_copies INT NOT NULL,

    FOREIGN KEY(Book_Id) REFERENCES BOOK(Book_Id),
    FOREIGN KEY(Branch_Id) REFERENCES LIBRARY_BRANCH(Branch_Id)
);
```

TASK 2

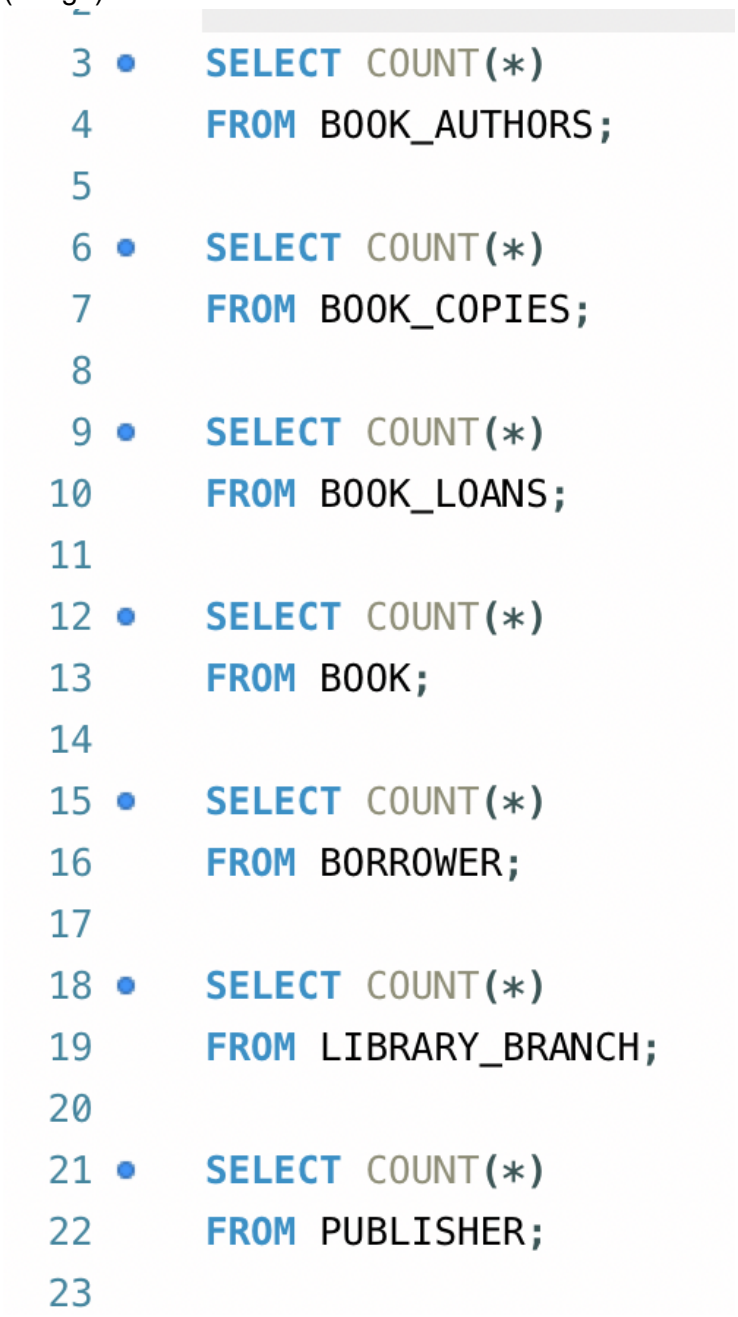
```
.separator ,
.import Book_Authors.csv BOOK_AUTHORS --skip 1;
.import Book_Copies.csv BOOK_COPIES --skip 1;
.import Book_Loans.csv BOOK_LOANS --skip 1;
.import Book.csv BOOK --skip 1;
.import Borrower.csv BORROWER --skip 1;
.import Library_Branch.csv LIBRARY_BRANCH --skip 1;
.import Publisher.csv PUBLISHER --skip 1;
```

LOADING/INSERTING THE DATA FROM TH CSV:
Use .separator , to read values from the CSV file
Use .import csv_file_name.csv TABLE_NAME --skip 1;
(skip 1 used to skip the header of the CSV file as that is not data)
All the CSV files were in the same folder as my Database

We used a .sql file which had the above command and then using the .read command on that sql file using sqlite3

Counts of all Inserted Data:
SQL Commands to get the count of each table:

(Image)



```
SELECT COUNT(*)
FROM BOOK_AUTHORS;

SELECT COUNT(*)
FROM BOOK_COPIES;

SELECT COUNT(*)
FROM BOOK_LOANS;

SELECT COUNT(*)
FROM BOOK;

SELECT COUNT(*)
FROM BORROWER;

SELECT COUNT(*)
FROM LIBRARY_BRANCH;

SELECT COUNT(*)
FROM PUBLISHER;
```

(Text)

```
SELECT COUNT(*)
FROM BOOK_AUTHORS;

SELECT COUNT(*)
FROM BOOK_COPIES;
```

```
SELECT COUNT(*)
FROM BOOK_LOANS;

SELECT COUNT(*)
FROM BOOK;

SELECT COUNT(*)
FROM BORROWER;

SELECT COUNT(*)
FROM LIBRARY_BRANCH;

SELECT COUNT(*)
FROM PUBLISHER;
```

The result of Count of each table in the sequence shown above:

```
+----------+
| COUNT(*) |
+----------+
| 21       |
+----------+
+----------+
| COUNT(*) |
+----------+
| 21       |
+----------+
+----------+
| COUNT(*) |
+----------+
| 21       |
+----------+
+----------+
| COUNT(*) |
+----------+
| 21       |
+----------+
+----------+
| COUNT(*) |
+----------+
| 21       |
+----------+
+----------+
| COUNT(*) |
+----------+
| 3        |
+----------+
+----------+
| COUNT(*) |
+----------+
| 17       |
```

The result shows that the PUBLISHER table has 17 entries and LIBRARY_BRANCH has 3 entries. All other tables have 21 entries.

All Inserted Data shown on Terminal:

Book_id	Author_name
1	Harper Lee
2	George Orwell
3	Jane Austen
4	F. Scott Fitzgerald
5	Gabriel Garcia Marquez
6	George Orwell
7	J.D. Salinger
8	William Golding
9	Aldous Huxley
10	Oscar Wilde
11	Paulo Coelho
12	Arundhati Roy
13	Emily Bronte
14	J.R.R. Tolkien
15	J.R.R. Tolkien
16	Douglas Adams
17	Anne Frank
18	Dan Brown
19	Mark Twain
20	Mark Twain
21	Charles Dickens

Book_Id	Branch_Id	No_of_copies
1	1	3
2	1	2
3	2	1
4	3	4
5	1	5
6	2	3
7	2	2
8	3	1
9	1	4
10	2	2
11	1	3
12	3	2
13	3	1
14	1	5
15	3	1
16	2	3
17	3	2
18	3	2
19	1	5
20	3	1
21	3	1

Book_ID	Branch_Id	Card_No	Date_Out	Due_Date	Returned_date
1	1	123456	2022-01-01	2022-02-01	2022-02-01
2	1	789012	2022-01-02	2022-02-02	NULL
3	2	345678	2022-01-03	2022-02-03	NULL
4	3	901234	2022-01-04	2022-02-04	2022-02-04
5	1	567890	2022-01-05	2022-02-05	2022-02-09
6	2	234567	2022-01-06	2022-02-06	2022-02-10
7	2	890123	2022-01-07	2022-02-07	2022-03-08
8	3	456789	2022-01-08	2022-02-08	2022-03-10
9	1	111111	2022-01-09	2022-02-09	2022-02-06
10	2	222222	2022-01-10	2022-02-10	2022-02-07

Book_Id	Title	Publisher_name
1	To Kill a Mockingbird	HarperCollins
2	1984	Penguin Books
3	Pride and Prejudice	Penguin Classics
4	The Great Gatsby	Scribner
5	One Hundred Years of Solitude	Harper & Row
6	Animal Farm	Penguin Books
7	The Catcher in the Rye	Little, Brown and Company
8	Lord of the Flies	Faber and Faber
9	Brave New World	Chatto & Windus
10	The Picture of Dorian Gray	Ward, Lock and Co.
11	The Alchemist	HarperCollins
12	The God of Small Things	Random House India
13	Wuthering Heights	Thomas Cautley Newby
14	The Hobbit	Allen & Unwin
15	The Lord of the Rings	Allen & Unwin
16	The Hitchhiker's Guide to the Galaxy	Pan Books
17	The Diary of a Young Girl	Bantam Books
18	The Da Vinci Code	Doubleday
19	The Adventures of Huckleberry Finn	Penguin Classics
20	The Adventures of Tom Sawyer	American Publishing Company
21	A Tale of Two Cities	Chapman and Hall

Card_No	Name	Address	Phone
111111	Alex Kim	983 Sine St, Arizona, AR 70451	678-784-5563
121212	Chloe Park	345 Shark St, Arizona, AR 72213	755-905-5572
123456	John Smith	456 Oak St, Arizona, AR 70010	205-555-5555
222222	Rachel Lee	999 Apple Ave, Arizona, AR 70671	231-875-5564
232323	William Chen	890 Sting St, New York, NY 10459	406-755-5580
234567	Emily Lee	389 Oaklay St, Arizona, AR 70986	231-678-5560
333333	William Johnson	705 Paster St, New Jersey 32002	235-525-5567
343434	Olivia Johnson	345 Pine St, New Jersey, NJ 32095	662-554-5575
345678	Bob Johnson	12 Elm St, Arizona, AR 70345	545-234-5557
444444	Ethan Martinez	466 Deeplm St, New York, NY 10321	555-555-5569
454545	Dylan Kim	567 Cowboy way St, New Jersey, NJ 32984	435-254-5578
456789	Laura Chen	345 Mapman Ave, Arizona, AR 70776	565-985-9962
555555	Grace Hernandez	315 Babes St, Arizona, AR 70862	455-567-5587
565656	Sophia Park	678 Dolphin St, New York, NY 10062	675-455-5568
567890	Tom Lee	678 S Oak St, New York, NY 10045	209-525-5559
676767	Olivia Lee	345 Spine St, New York, NY 10092	435-878-5569
787878	Noah Thompson	189 GreenOak Ave, New Jersey, NJ 32453	245-555-5571
789012	Jane Doe	789 Maple Ave, New Jersey, NJ 32542	555-235-5556
890123	Michael Park	123 Pinewood St, New Jersey, NJ 32954	655-890-2161
901234	Sarah Kim	345 Pine St, New York, NY 10065	515-325-2158
989898	Olivia Smith	178 Elm St, New Jersey, NJ 32124	325-500-5579

Branch_Id	Branch_Name	Branch_Address
1	Main Branch	123 Main St, New York, NY 10003
2	West Branch	456 West St, Arizona, AR 70622
3	East Branch	789 East St, New Jersy, NY 32032

Publisher_name	Phone	Address
HarperCollins	212-207-7000	195 Broadway, New York, NY 10007
Penguin Books	212-366-3000	475 Hudson St, New York, NY 10014
Penguin Classics	212-366-2000	123 Main St, California, CA 01383
Scribner	212-207-7474	19 Broadway, New York, NY 10007
Harper & Row	212-207-7000	1195 Border street, Montana, MT 59007
Little, Brown and Company	212-764-2000	111 Huddle St, New Jersey, NJ 32014
Faber and Faber	201-797-3800	463 south centre street, Arizona, AR 71653
Chatto & Windus	442-727-3800	Bloomsbury House, 7477 Great Russell St, Arizona, AR 72965
Ward, Lock and Co.	647-242-3434	456 Maple Ave, Texas, TX 76013
Random House India	291-225-6634	423 baywatch centre street, Alabama, AL 30513
Thomas Cautley Newby	243-353-2352	890 Elmwood Dr, Floride, FL 98238
Allen & Unwin	212-782-9001	22 New Wharf Rd, Arizona, AR 70654
Pan Books	313-243-5353	567 Pine Tree Rd, Colorado, CO 87348
Bantam Books	313-243-5354	1745 Broadway, New York, NY 10019
Doubleday	212-782-9000	789 Division St, Minnesota, MN 55344
American Publishing Company	682-243-3524	7652 Northgate way lane, Georgia, GA 30054
Chapman and Hall	833-342-2343	789 Oak St, Texas, TX 76010

```
qlite>
```

Challenges: There weren't any major challenges. The tasks can be tedious. But using SQL files and using methods similar to Project 1 helped save a lot of time as we ran sql once to run multiple commands. We also matched values we got through SQL with the CSV file to ensure they were properly loaded.

TASK 3

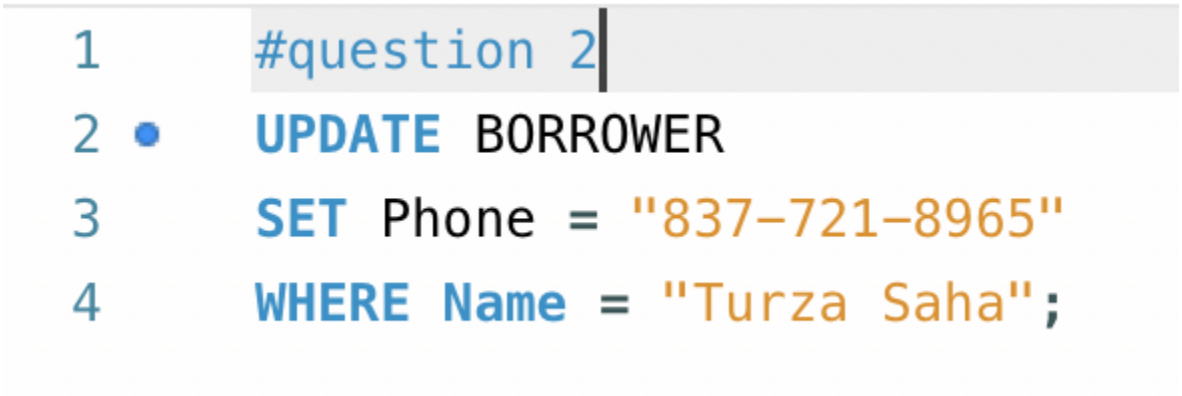
Question 1

```
3    #Question 1
4 •  INSERT INTO BORROWER(Name, Address, Phone) VALUES( "Turza Saha", "400 Kerby Street", "682-812-355");
```

INSERT INTO BORROWER(Name, Address, Phone) VALUES("Turza Saha", "400 Kerby Street", "682-812-355");

Assumption: We assume the Card_No is auto incremented as we did not insert the Card_No

Question 2



UPDATE BORROWER
SET Phone = "837-721-8965"
WHERE Name = "Turza Saha";

Result after Question 2: Shows that new borrower added with auto incremented Card_no and updated Phone_number, 1 row affected

```
sqlite> SELECT * FROM BORROWER
   ...> ;
```

Card_No	Name	Address	Phone
111111	Alex Kim	983 Sine St, Arizona, AR 70451	678-784-5563
121212	Chloe Park	345 Shark St, Arizona, AR 72213	755-905-5572
123456	John Smith	456 Oak St, Arizona, AR 70010	205-555-5555
222222	Rachel Lee	999 Apple Ave, Arizona, AR 70671	231-875-5564
232323	William Chen	890 Sting St, New York, NY 10459	406-755-5580
234567	Emily Lee	389 Oaklay St, Arizona, AR 70986	231-678-5560
333333	William Johnson	705 Paster St, New Jersey 32002	235-525-5567
343434	Olivia Johnson	345 Pine St, New Jersey, NJ 32095	662-554-5575
345678	Bob Johnson	12 Elm St, Arizona, AR 70345	545-234-5557
444444	Ethan Martinez	466 Deeplm St, New York, NY 10321	555-555-5569
454545	Dylan Kim	567 Cowboy way St, New Jersey, NJ 32984	435-254-5578
456789	Laura Chen	345 Mapman Ave, Arizona, AR 70776	565-985-9962
555555	Grace Hernandez	315 Babes St, Arizona, AR 70862	455-567-5587
565656	Sophia Park	678 Dolphin St, New York, NY 10062	675-455-5568
567890	Tom Lee	678 S Oak St, New York, NY 10045	209-525-5559
676767	Olivia Lee	345 Spine St, New York, NY 10092	435-878-5569
787878	Noah Thompson	189 GreenOak Ave, New Jersey, NJ 32453	245-555-5571
789012	Jane Doe	789 Maple Ave, New Jersey, NJ 32542	555-235-5556
890123	Michael Park	123 Pinewood St, New Jersey, NJ 32954	655-890-2161
901234	Sarah Kim	345 Pine St, New York, NY 10065	515-325-2158
989898	Olivia Smith	178 Elm St, New Jersey, NJ 32124	325-500-5579
989899	Turza Saha	400 Kerby Street	837-721-8965

Question 3

```
1  UPDATE BOOK_COPIES
2  SET No_Of_Copies = No_Of_Copies +1
3  WHERE Branch_ID IN(
4          SELECT Branch_Id
5          FROM LIBRARY_BRANCH
6          WHERE Branch_Name = "East Branch"
7      );
```

```
UPDATE BOOK_COPIES
SET No_Of_Copies = No_Of_Copies +1
WHERE Branch_ID IN(
                SELECT Branch_Id
      FROM LIBRARY_BRANCH
      WHERE Branch_Name = "East Branch"
        );
```

Result: You can see that there is one more book copy in each row with Branch_Id = 3 compared to initial CSV file, 9 rows affected

```
                "--- error here
[sqlite> SELECT * FROM BOOK_COPIES;
+---------+-----------+--------------+
| Book_Id | Branch_Id | No_of_copies |
+---------+-----------+--------------+
| 1       | 1         | 3            |
| 2       | 1         | 2            |
| 3       | 2         | 1            |
| 4       | 3         | 5            |
| 5       | 1         | 5            |
| 6       | 2         | 3            |
| 7       | 2         | 2            |
| 8       | 3         | 2            |
| 9       | 1         | 4            |
| 10      | 2         | 2            |
| 11      | 1         | 3            |
| 12      | 3         | 3            |
| 13      | 3         | 2            |
| 14      | 1         | 5            |
| 15      | 3         | 2            |
| 16      | 2         | 3            |
| 17      | 3         | 3            |
| 18      | 3         | 3            |
| 19      | 1         | 5            |
| 20      | 3         | 2            |
| 21      | 3         | 2            |
+---------+-----------+--------------+
```

Question 4a:

```
2 •  INSERT INTO BOOK(Title, Publisher_Name) VALUES("Harry Potter and the Sorcerer\'s Stone","Oxford Publisheing");
3 •  INSERT INTO BOOK_AUTHORS(Book_Id, Author_Name) VALUES ((SELECT Book_Id FROM BOOK WHERE Title = "Harry Potter and the Sorcerer\'s Stone"),
4        "J.K. Rowling");
```

INSERT INTO BOOK(Title, Publisher_Name) VALUES("Harry Potter and the Sorcerer\'s Stone","Oxford Publisheing");
INSERT INTO BOOK_AUTHORS(Book_Id, Author_Name) VALUES ((SELECT Book_Id FROM BOOK WHERE Title = "Harry Potter and the Sorcerer\'s Stone"),

"J.K. Rowling");

Assumption: The Book_ID of the new inserted book is auto-incremented and the same Book_ID is selected as a foreign key for the second insert statement into BOOK_AUTHORS.

Result:
It shows how the new book was added with auto-incremented Book_Id, 1 row affected

```
sqlite> SELECT * FROM BOOK;
+---------+----------------------------------------+-----------------------------+
| Book_Id |                 Title                  |       Publisher_name        |
+---------+----------------------------------------+-----------------------------+
| 1       | To Kill a Mockingbird                  | HarperCollins               |
| 2       | 1984                                   | Penguin Books               |
| 3       | Pride and Prejudice                    | Penguin Classics            |
| 4       | The Great Gatsby                       | Scribner                    |
| 5       | One Hundred Years of Solitude          | Harper & Row                |
| 6       | Animal Farm                            | Penguin Books               |
| 7       | The Catcher in the Rye                 | Little, Brown and Company   |
| 8       | Lord of the Flies                      | Faber and Faber             |
| 9       | Brave New World                        | Chatto & Windus             |
| 10      | The Picture of Dorian Gray             | Ward, Lock and Co.          |
| 11      | The Alchemist                          | HarperCollins               |
| 12      | The God of Small Things                | Random House India          |
| 13      | Wuthering Heights                      | Thomas Cautley Newby        |
| 14      | The Hobbit                             | Allen & Unwin               |
| 15      | The Lord of the Rings                  | Allen & Unwin               |
| 16      | The Hitchhiker's Guide to the Galaxy   | Pan Books                   |
| 17      | The Diary of a Young Girl              | Bantam Books                |
| 18      | The Da Vinci Code                      | Doubleday                   |
| 19      | The Adventures of Huckleberry Finn     | Penguin Classics            |
| 20      | The Adventures of Tom Sawyer           | American Publishing Company |
| 21      | A Tale of Two Cities                   | Chapman and Hall            |
| 22      | Harry Potter and the Sorcerer\'s Stone | Oxford Publisheing          |
+---------+----------------------------------------+-----------------------------+
```

Result: The result of the 2nd insert statement using the last book_id in BOOK, 1 row affected

```
sqlite> SELECT * FROM BOOK_AUTHORS;
+---------+----------------------+
| Book_id |      Author_name     |
+---------+----------------------+
| 1       | Harper Lee           |
| 2       | George Orwell        |
| 3       | Jane Austen          |
| 4       | F. Scott Fitzgerald  |
| 5       | Gabriel Garcia Marquez |
| 6       | George Orwell        |
| 7       | J.D. Salinger        |
| 8       | William Golding      |
| 9       | Aldous Huxley        |
| 10      | Oscar Wilde          |
| 11      | Paulo Coelho         |
| 12      | Arundhati Roy        |
| 13      | Emily Bronte         |
| 14      | J.R.R. Tolkien       |
| 15      | J.R.R. Tolkien       |
| 16      | Douglas Adams        |
| 17      | Anne Frank           |
| 18      | Dan Brown            |
| 19      | Mark Twain           |
| 20      | Mark Twain           |
| 21      | Charles Dickens      |
| 22      | J.K. Rowling         |
+---------+----------------------+
```

Question 4b

```
1
2 •  INSERT INTO LIBRARY_BRANCH(Branch_Name, Branch_Address) VALUES("North Branch","456 NW, Irving, TX 76100");
3 •  INSERT INTO LIBRARY_BRANCH(Branch_Name, Branch_Address) VALUES("UTA Branch", "123 Cooper St, Arlington TX 76101");
4
```

INSERT INTO LIBRARY_BRANCH(Branch_Name, Branch_Address) VALUES("North Branch","456 NW, Irving, TX 76100");
INSERT INTO LIBRARY_BRANCH(Branch_Name, Branch_Address) VALUES("UTA Branch", "123 Cooper St, Arlington TX 76101");

Assumption: It is assumed that the Branch_Id is auto incremented even if not directly inserted.

Result: Shows how Branch_Id was auto incremented, 2 rows affected as 2 rows were added

```
sqlite> .read Query4b.sql
sqlite> SELECT * FROM LIBRARY_BRANCH;
+-----------+-------------+-----------------------------------+
| Branch_Id | Branch_Name |          Branch_Address           |
+-----------+-------------+-----------------------------------+
| 1         | Main Branch | 123 Main St, New York, NY 10003   |
| 2         | West Branch | 456 West St, Arizona, AR 70622    |
| 3         | East Branch | 789 East St, New Jersy, NY 32032  |
| 4         | North Branch| 456 NW, Irving, TX 76100          |
| 5         | UTA Branch  | 123 Cooper St, Arlington TX 76101 |
+-----------+-------------+-----------------------------------+
sqlite>
```

Question 5

```
SELECT B.Title, LB.Branch_Name,(STRFTIME('%Y%m%d', BL.Returned_date)-STRFTIME('%Y%m%d',BL.Date_Out )) as DaysBorrowed
FROM BOOK as B
JOIN BOOK_LOANS BL ON BL.Book_ID = B.Book_Id
JOIN LIBRARY_BRANCH LB on LB.Branch_Id = BL.Branch_Id
WHERE STRFTIME('%Y%m%d',BL.Date_Out) BETWEEN STRFTIME('%Y%m%d', date('2022-03-05'))
AND STRFTIME('%Y%m%d', date('2022-03-23'));
```

SELECT B.Title, LB.Branch_Name,(STRFTIME('%Y%m%d', BL.Returned_date)-STRFTIME('%Y%m%d',BL.Date_Out)) as DaysBorrowed
FROM BOOK as B
JOIN BOOK_LOANS BL ON BL.Book_ID = B.Book_Id
JOIN LIBRARY_BRANCH LB on LB.Branch_Id = BL.Branch_Id
WHERE STRFTIME('%Y%m%d',BL.Date_Out) BETWEEN STRFTIME('%Y%m%d', date('2022-03-05'))
AND STRFTIME('%Y%m%d', date('2022-03-23'));

We used STRFTIME to work with Date formatted Date to use BETWEEN and to use the difference of date as DaysBorrowed.

Result: 2 rows affected

```
sqlite> .read Query5.sql
+---------------------------------------+-------------+--------------+
|                Title                  | Branch_Name | DaysBorrowed |
+---------------------------------------+-------------+--------------+
| The Hitchhiker's Guide to the Galaxy  | West Branch | 19           |
| The Diary of a Young Girl             | East Branch | 7            |
+---------------------------------------+-------------+--------------+
```

Question 6

```sql
SELECT B.Name
FROM Borrower as B
JOIN BOOK_LOANS BL ON BL.Card_No = B.Card_No
WHERE BL.Returned_date = "NULL";
```

SELECT B.Name
FROM Borrower as B
JOIN BOOK_LOANS BL ON BL.Card_No = B.Card_No
WHERE BL.Returned_date = "NULL";

Assumption: In the dataset, instead of nothing or empty, any unreturned book has “NULL” as value in its Returned_date

Result: 2 rows affected

```
sqlite> .read Query6.sql
+--------------+
|     Name     |
+--------------+
| Jane Doe     |
| Bob Johnson  |
+--------------+
```

Question 7

```sql
SELECT LB.Branch_Name, SUM(BL.Returned_Date = "NULL") as StillBorrowed, SUM(BL.Returned_Date<>"NULL") as BookReturned,
SUM(BL.Returned_date>BL.Due_Date) as ReturnedLate
FROM LIBRARY_BRANCH as LB
JOIN BOOK_LOANS BL ON BL.Branch_Id = LB.Branch_Id
GROUP BY LB.Branch_Name;
```

SELECT LB.Branch_Name, SUM(BL.Returned_Date = "NULL") as StillBorrowed,
SUM(BL.Returned_Date<>"NULL") as BookReturned,
SUM(BL.Returned_date>BL.Due_Date) as ReturnedLate
FROM LIBRARY_BRANCH as LB
JOIN BOOK_LOANS BL ON BL.Branch_Id = LB.Branch_Id
GROUP BY LB.Branch_Name;

Assumption: If “NULL” Returned_date, then book is still borrowed, if not “NULL” then Book has been returned. And Book returned late is also considered as returned. Book Returned_Date later than Due_Date is counted as late, so any book returned ON the return_date is not late.

Result: 3 rows affected

```
sqlite> .read Query7.sql
+-------------+---------------+--------------+--------------+
| Branch_Name | StillBorrowed | BookReturned | ReturnedLate |
+-------------+---------------+--------------+--------------+
| East Branch | 0             | 9            | 2            |
| Main Branch | 1             | 6            | 3            |
| West Branch | 1             | 4            | 4            |
+-------------+---------------+--------------+--------------+
```

Question 8

```
3 •   SELECT B.Title, MAX(julianday(BL.Returned_date)-julianday(BL.Date_Out)) as MaximumDaysBorrowed
4     FROM BOOK as B
5     JOIN BOOK_LOANS BL ON BL.Book_ID = B.Book_Id
6     WHERE BL.Returned_date <> "NULL"
7     GROUP BY Title
8     ORDER BY MaximumDaysBorrowed DESC;
```

SELECT B.Title, MAX(julianday(BL.Returned_date)-julianday(BL.Date_Out)) as MaximumDaysBorrowed
FROM BOOK as B
JOIN BOOK_LOANS BL ON BL.Book_ID = B.Book_Id
WHERE BL.Returned_date <> "NULL"
GROUP BY Title
ORDER BY MaximumDaysBorrowed DESC;

Assumption/ Challenge: strftime did not return the right number of days. Like a book that had 31 Days MaximumDaysBorrowed showed 100
Thus, we used julianday which worked and got the exact days, as we also matched with the actual data, although the answer is in float.
We are also not taking into account Books that have not been returned date into consideration for MaximumDaysBorrowed
Also, additional use of ORDER BY MaximumdaysBorrowed DESC just to give a better structure to the result below.

Result: 19 rows affected

```
sqlite> .read Query8.sql
```

Title	MaximumDaysBorrowed
The Hobbit	76.0
Lord of the Flies	61.0
The Catcher in the Rye	60.0
The Lord of the Rings	37.0
One Hundred Years of Solitude	35.0
Animal Farm	35.0
To Kill a Mockingbird	31.0
The Great Gatsby	31.0
The Da Vinci Code	31.0
The Adventures of Tom Sawyer	31.0
A Tale of Two Cities	31.0
The Picture of Dorian Gray	28.0
Brave New World	28.0
The Hitchhiker's Guide to the Galaxy	19.0
Wuthering Heights	15.0
The God of Small Things	7.0
The Diary of a Young Girl	7.0
The Alchemist	7.0
The Adventures of Huckleberry Finn	7.0

Question 9

```
3 •   SELECT BR.Name,B.Title, BA.Author_Name,
4     (julianday(BL.Returned_date)-julianday(BL.Date_Out)) as DaysBorrowed,
5     (CASE WHEN BL.Returned_date>Bl.Due_Date THEN "Returned Late" ELSE "Returned On Time" END) as ReturnStatus
6     FROM BORROWER as BR
7     JOIN BOOK_LOANS BL ON BL.Card_No = BR.Card_No
8     JOIN BOOK B ON B.Book_Id = BL.Book_ID
9     JOIN BOOK_AUTHORS BA ON BA.Book_Id = BL.Book_ID
10    WHERE BR.Name = "Ethan Martinez"
11    ORDER BY BL.date_out;
```

SELECT BR.Name,B.Title, BA.Author_Name,
(julianday(BL.Returned_date)-julianday(BL.Date_Out)) as DaysBorrowed,
(CASE WHEN BL.Returned_date>Bl.Due_Date THEN "Returned Late" ELSE "Returned On Time" END) as ReturnStatus
FROM BORROWER as BR

```
JOIN BOOK_LOANS BL ON BL.Card_No = BR.Card_No
JOIN BOOK B ON B.Book_Id = BL.Book_ID
JOIN BOOK_AUTHORS BA ON BA.Book_Id = BL.Book_ID
WHERE BR.Name = "Ethan Martinez"
ORDER BY BL.date_out;
```

Assumption/Challenges: Once again we use juliandays to get exact number of days Book was Borrowed
We are assuming that the Returned_Date value is not NULL for Ethan Martinez, ie we assume Ethan did return the book. Or else, the result may come as “Returned On Time”

```
sqlite> .read Query10.sql
+----------------+-------------------------+---------------+--------------+------------------+
|      Name      |          Title          |  Author_name  | DaysBorrowed |   ReturnStatus   |
+----------------+-------------------------+---------------+--------------+------------------+
| Ethan Martinez | The God of Small Things | Arundhati Roy | 7.0          | Returned On Time |
+----------------+-------------------------+---------------+--------------+------------------+
```

Query 10

```
1
2  SELECT BR.Name, BR.Address
3  FROM BORROWER as BR
4  JOIN BOOK_LOANS BL ON BL.Card_No = BR.Card_No
5  JOIN LIBRARY_BRANCH LB ON LB.Branch_Id = BL.Branch_Id
6  WHERE LB.Branch_Name = "West Branch";
```

```
SELECT BR.Name, BR.Address
FROM BORROWER as BR
JOIN BOOK_LOANS BL ON BL.Card_No = BR.Card_No
JOIN LIBRARY_BRANCH LB ON LB.Branch_Id = BL.Branch_Id
WHERE LB.Branch_Name = "West Branch";
```

```
false error near line 2: no such column: B.card_No
sqlite> .read Query10.sql
+---------------+----------------------------------------+
|     Name      |                Address                 |
+---------------+----------------------------------------+
| Bob Johnson   | 12 Elm St, Arizona, AR 70345           |
| Emily Lee     | 389 Oaklay St, Arizona, AR 70986       |
| Michael Park  | 123 Pinewood St, New Jersey, NJ 32954  |
| Rachel Lee    | 999 Apple Ave, Arizona, AR 70671       |
| Noah Thompson | 189 GreenOak Ave, New Jersey, NJ 32453 |
+---------------+----------------------------------------+
```
