Week 7 classwork

For lectures on 16th and 17th of November

Recall the following schema from Problem Class 2.

Book (ISBN, title, author)
Fiction (ISBN, numpages, genre, ispaperback, price)
NonFiction(ISBN, numpages, genre, ispaperback, istextbook, price)
Periodical (ISBN, numpages, seriesno, issueno, price)

In that problem class we constructed some relational algebra queries on this database. Today we will write SQL statements to answer some of the same and some new queries. Before we can do that however we must create the four tables and insert some data into them.

(0) Create a database called bookdb.db in your SQLite folder.
This may be done from the Command Prompt by first navigating to the SQLite folder on your machine, then writing:
> *sqlite3 bookdb.db*

After this, you will enter the SQLite command prompt where you can type SQL statements.

In order to save a new copy of your database, write:

> *.save bookdb_new.db*

To exit the database, write:
>*.quit*

(exiting with .quit should autosave the database).

(1) We first create tables **Book**, **Fiction**, **NonFiction** and **Periodical**. We need to make sure we choose the right data types for each attribute. We must also specify the primary keys and foreign keys. Note that ISBN is the primary key in each table. However, in Fiction, NonFiction and Periodical it is also the foreign key referencing the ISBN in Book. Note that in SQLite foreign key constraints are not enforced by default. Enter the following command to enforce them:

> *PRAGMA foreign_keys=1;*

Run the following commands to create the Book and Fiction table”

```
CREATE TABLE Book(ISBN VARCHAR(20) PRIMARY KEY, title VARCHAR(100), author
VARCHAR(50));

CREATE TABLE Fiction(ISBN VARCHAR(20) PRIMARY KEY, numpages INTEGER,
genre VARCHAR(10), ispaperback INTEGER, price REAL,
FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
);
```

Similarly, create the **NonFiction** and **Periodical** tables.

(i) Enter the *.tables* command to display the list of all tables.
(ii) Enter the *.schema* command to display the system schema.
(iii) Enter the *.schema <table name>* to display the schema of a specific table.
(iv) To clear the screen at any time, use >*.shell cls* (.shell clear on Mac's terminal)

(2) Now that we have the tables in place, we need to insert some data into them. Insert one row in the Book table and one row in the fiction table only. Note that for Boolean types, SQLite uses literals 0 and 1. You may add data for the book Nineteen Eighty-Four by George Orwell. For ISBN use 978-0-15-565811-0. It is a fiction paperback with 328 pages. For genre, use Dystopian/Political. For price use 2.5 (units are always pounds).

Recall the general format of the INSERT statement:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

(3) Read the script from the file books_data.sql into your database. You may do this from the SQLite command prompt by writing >*.read books_data.sql*. Make sure the file is available in your SQLite folder. Now the database should have some more rows to work with. Write the SELECT * table_name; statement for each table to look at the data in it.

(4) Write DQL statements to answer the following queries on this data. (Results have been provided for confirmation).

For custom column name use the aliasing keyword AS after the column name. For example, SELECT name AS 'Employee Name' from Employee;
You can also give aliases to table names by using the AS keyword after the table name.

Note: to see the column headers in the query results, first run the commands >*.headers on* as header names are not shown by default in SQLite.

(a) List the ISBNs of all periodicals.

```
ISBN
953-5-23-378610-0
902-1-98-334410-0
953-7-21-334410-0
```

(b) Print the ISBNs of all periodicals with less than 200 pages.

```
ISBN
902-1-98-334410-0
953-7-21-334410-0
```

(c) List the titles and authors of all fiction books in the system.

```
title|author
Nineteen Eighty-four|George Orwell
Crime and Punishment|Fyodor Dostoevsky
Norwegian Wood|Haruki Murakami
The Garlic Ballads|Mo Yan
Slaughterhouse-Five|Kurt Vonnegut
```

(d) List the titles and authors of all non-fiction text books not available in paperback.

```
title|author
A Short History of Europe|Simon Jenkins
```

(e) List the titles and authors of all fiction and non-fiction books in the system only (not periodicals).

```
title|author
A Short History of Europe|Simon Jenkins
Crime and Punishment|Fyodor Dostoevsky
Nineteen Eighty-four|George Orwell
Norwegian Wood|Haruki Murakami
Sapiens: A Brief History of Humankind|Yuval Noah Harari
Slaughterhouse-Five|Kurt Vonnegut
Superintelligence|Nick Bostrom
The Garlic Ballads|Mo Yan
```

(f) Compute the total number of books in the system. Display the count under the header 'No of Books in the System'

```
No of Books in the System
11
```

(g) What is the average length of fiction books in the system?

```
Avg fiction book length
382.4
```

(h) What are the numbers of fiction books in different genres?

```
genre|Num of books
Drama|2
Dystopian/Political|1
Historical|1
War|1
```

(i) Which authors have written books (of any kind) longer than 500 pages?

```
author
Fyodor Dostoevsky
Mo Yan
```

(j) Run the following queries one by one. What are the answers and how do you explain them?
```
SELECT COUNT(*) from Book;
SELECT COUNT(*) from Fiction;
SELECT COUNT(*) from NonFiction;
Select COUNT(*) from Book, Fiction;
Select COUNT(*) from Book, Fiction, Nonfiction;
```

(k) Run the following queries on by one. What are the answers and how do you explain them?
```
Select COUNT(*) from Book NATURAL JOIN Fiction;
Select COUNT(*) from Book NATURAL JOIN Fiction NATURAL JOIN NonFiction;
```

(l) What are the outputs of the following two queries:
```
Select * from Book NATURAL JOIN Fiction;
Select * from Book INNER JOIN Fiction ON Book.ISBN=Fiction.ISBN;
```

(m) Run the following query and observe the results. How do you explain these results:
```
select author from book, fiction, nonfiction where (book.ISBN=fiction.ISBN AND
fiction.numpages>=500);
```

(n) Run the following queries and observe their results. What is the effect of the ORDER BY clause in each case?
```
Select genre, COUNT(ISBN) from Book NATURAL JOIN Fiction GROUP by genre ORDER by
count(ISBN);
Select genre, COUNT(ISBN) from Book NATURAL JOIN Fiction GROUP by genre ORDER by
genre;
```

Also try:

Select genre, COUNT(ISBN) from Book NATURAL JOIN Fiction GROUP by genre ORDER by count(ISBN) DESC;

(o) Run the following query and observe the result. What is the effect of the HAVING clause in this case:

Select genre, COUNT(ISBN) from Book NATURAL JOIN Fiction GROUP by genre HAVING numpages<300;

***