GRADED LAB

CITS1402 Relational Database Management Systems – Lab 2
Released on 13 March
Due date 26 March, Tuesday at 1800 hrs (6pm)
How to submit: On LMS. UC will show the submission link during the class on 14 March, in case you cannot find it. The questions are listed on page 4.
Marks: 4% of the total grade
Use of AI: You are allowed to consult each other or get help from SQL tutorials. AI (LLMs) such as ChatGPT are not allowed.

You will study the topics required to complete this lab in *weeks 3 & 4*.

Learning Aims
This lab is concerned with
1. Creating and populating an SQLite table
2. Examining the table and its entries
3. Basic use of the SELECT statement and WHERE clause

Case study:

SafePrint Solutions is a small printing company that works for book publishers (tracked in the **publishers** table). SafePrint Solutions' jobs consist of printing books or parts of books. These jobs are recorded in the **bookjobs** table. A printing job requires the use of materials, such as paper and ink, which are assigned to a job via purchase orders (PO) kept in the **pos** table. Each printing job may have several POs assigned to it. Likewise, each PO may contain several PO items which are recorded in a separate **po_items** table. The one-to-many relationship between **pos** and **po_items** is implemented by the composite foreign key (*job_id, po_id*) in the **po_items** table. The materials which appear in **po_items** are tracked in the **items** table, which records the material description, the quantity on-hand in the warehouse, and the price.

The many-to-many (*:*) relationship between **pos** and **items** is decomposed into two one-to-many (1:*) relationships by means of the intersection relation **po_items**. [will be taught in week 4]

SafePrint Solutions Relation Structures

publishers (cust_id, name, city, phone, creditcode)

bookjobs (job_id, cust_id, job_date, descr, jobtype)

pos (job_id, po_id, po_date, vendor_id)
items (item_id, descr, on_hand, price)

po_items (job_id, po_id, item_id, quantity)

Creating the database

You are provided with 3 files, that will help you load the database.

There are multiple ways to load a database in your sqlite. Let's look at those ways

1. Loading a .db file
   You need to open a windows or a mac terminal. Now navigate to the folder where you have downloaded the files that we have provided. You can do that using the cd command. In order to display the contents of the folder please type dir (on windows) or ls (on mac). Once you are in the right folder on the terminal please type sqlite3. Then load the database using the **.open** command as shown below.

```
mehwishnasim@Mehwishs-Mac-Studio Downloads % sqlite3
SQLite version 3.37.0 2021-12-09 01:34:53
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .database
main: "" r/w
sqlite> .open printing.db
sqlite> .database
main: /Users/mehwishnasim/Downloads/printing.db r/w
sqlite>
```

2. You can also load the database when you are opening your sqlite3.
   First you need to quit sqlite.
   Now write **sqlite3 printing.db**
   It will load the database.

```
mehwishnasim@Mehwishs-Mac-Studio Downloads % sqlite3 printing.db
SQLite version 3.37.0 2021-12-09 01:34:53
Enter ".help" for usage hints.
sqlite> .tables
bookjobs    items       po_items    pos         publishers
```

3. There is also a third way. It is useful in situations where you have played around with the database and you want to load a fresh copy without quitting sqlite. Also, at times .db file is not available.

- You can construct the database through files that have a ".sql" or a ".txt" extension.
- The files that are provided to you are:
  - load-tables-sqlite.sql : this file will create tables. If you open this using a textpad (right-click and open with textpad or notepad), you will see multiple drop table statements. Those statements will drop any existing tables that have the same name. If you load this file in a fresh instance of sqlite, it will give you 5 errors. Why? Because the tables that you are trying to delete (DROP) do not exist. Ignore those errors.
  - Load it using the command **.read load-tables-sqlite.sql**

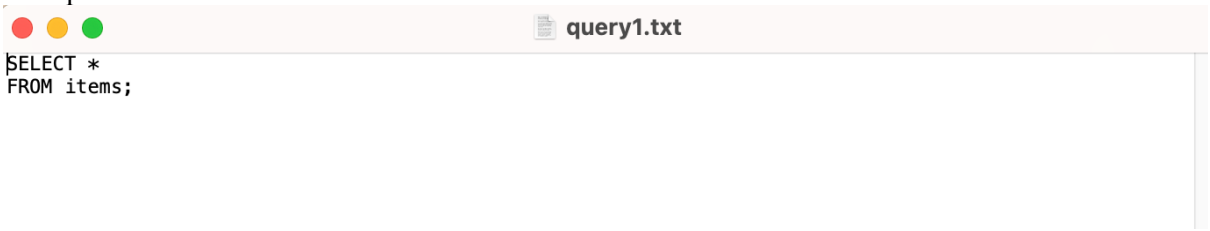
```
Error: near line 1: in prepare, no such table: po_items (1)
Error: near line 2: in prepare, no such table: items (1)
Error: near line 3: in prepare, no such table: pos (1)
Error: near line 4: in prepare, no such table: bookjobs (1)
Error: near line 5: in prepare, no such table: publishers (1)
sqlite> .tables
```

- Next load the data using **.read load-data.sql**
- Now type the **.scheme** command to see your relations and attributes.
- To retrieve data from a table type SELECT * FROM *tablename*;

Once you have the tables and data loaded, you can attempt the lab. Please note you can also save the database through the following command:
.save mydatabase.db (or any other name of your liking)

How should I submit my lab

For every question you need to create a .txt file. For example for Q1, create a file called Q1.txt. The file should only contain the relevant query and nothing else. For instance, see the following example:



How am I going to create a .txt file.
Windows:
You can create a .txt file on notepad. Make sure not to save it as .rtf or a .doc or a .pdf file. Also, a file format that is like "Q1.txt.rtf" is wrong. The reason is the files that have an extension other than .txt may introduce characters (invisible to you). For example, for a newline the character is "\n". You may not see it, but it is there. If the auto-grader sees those characters, it will give you zero points as it will not be able to run the query.

Mac/linux
Go to command prompt and type *vi query1.txt* (or Q1.txt etc.). Type your query. When finished, press the Esc key on the keyboard and then write :wq!

Verify as follows:

```
sqlite> .read query1.txt
P9|9KG PAPER|300|25.25
P12|12KG PAPER|700|49.99
P18|18KG PAPER|100|100
IRN|INK-RESIN|3|500
IWS|INK-WRSOL|5|350
CBD|CARDBOARD|47|15
```

You can verify all the answers, using the .read command one by one (for example ***.read Q2.txt*** or ***.read Q5.txt***)

Put all the queries in separate files: Q1.txt, Q2.txt … and so on, in a folder that is called "studentID_CITS1402_Lab2". Replace the student ID with your ID. For example, if your student ID is 123456, the folder name should look like: 123456_CITS1402_Lab2.

Zip the folder and submit it on LMS.

Your submission should run on our systems. If the file is corrupt or the naming structure is different, or if the file consists of irrelevant code or comments, the autograder will give you zero points.

The submission process sounds too overwhelming

No problem! Please see us during the lab hours or after the lecture/workshop to get help ☺

Write queries SQL queries for the following questions.

1. Write an SQL query that will list cust_id , name and phone of publisher where credit code is D.
2. Display an item's ID and price where the quantity on hand is less than 100.
3. Write an SQL query that will list the job Id and Po ID from the po items table. Ensure that either the quantity is greater than 50 or the item ID is IRN.
4. Find the unique job ID and po ID in the po items table. Ensure that either the quantity is greater than 50 or the item ID is IRN.
5. Display the minimum on_hand times price in a column called min and the maximum on_hand times price in a column called max, where the number of on_hand items is more than 100.
6. Write a SQL command that will output the average quantity in the po_items for all item IDs that are P9. The column should be named avg.
7. Display job_id, po_id, po_date, vendor_id from the pos table ensuring that the dates are not between January 01, 1990 and March 01, 1990
8. Display customer id and phone from the publishers table, for only those phone numbers that are either null or the last 4 digits do not contain number 6.