CSE 3241 Project Checkpoint 03 – SQL and More SQL

**Names** Maddison Chaffin, Defne Ceyhan, Chaeun Hong, Yi-Ting Tsan **Date:** 10/26/2022

As with all worksheets, these checkpoint parts must be submitted together on the Carmen Dropbox as a ZIP.

# Part One:

Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. **If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models.**

* BOOK(Isbn, Price, Title, Year, P\_name (FK))
* GENRE (B\_isbn (FK), Genre)
* AUTHOR(Id, Fname, Lname, Bdate)
* WAREHOUSE(Id, Location, Capacity)
* CUSTOMER(Username, Password, Email, Phone\_num)
* PUBLISHER (Name, Email)
* ORDERS(Order\_num, Order\_date, Ship\_address, Customer\_username(FK), Creditcard\_number(FK), Paypal\_username(FK))
* CREDIT\_CARD(Number, Company, Cvv, Name, Bill\_address)
* PAYPAL(Username, Password)
* STORES (B\_isbn(FK), W\_id(FK), Quantity )
* WRITES (B\_isbn(FK), A\_Id(FK))
* ADDED\_TO (B\_isbn(FK), O\_order\_num(FK), Quantity)
* REFUND(Order\_num(FK), Item, Count, Reason, Method)

# Part Two:

1. Given your relational schema, create a text file containing the SQL code to create your database schema. Use this SQL to create a database in SQLite. Populate this database with the data provided for the project as well as 20 sample records for each table that does not contain data provided in the original project documents.
2. Given your relational schema, provide the SQL to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named

“WorksheetTwoSimpleQueries.txt”:

* 1. Find the titles of all books by Pratchett that cost less than $10

SELECT Title

## FROM BOOK AS B, AUTHOR AS A, WRITES AS W

WHERE B.Isbn=W.B\_isbn AND A.Id=W.A\_Id AND A.Lname = 'Pratchett'

AND B.Price < 10;

* 1. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

SELECT B.Title, C.Username, O.Order\_date

## FROM BOOK AS B, CUSTOMER AS C, ADDED\_TO AS A, ORDERS AS O

WHERE B.Isbn=A.B\_isbn

AND A.O\_order\_num = O.Order\_num AND C.Username = O.Customer\_username AND C.Username = 'iVan2';

* 1. Find the titles and ISBNs for all books with less than 5 copies in stock SELECT B.Title, B.Isbn

## FROM BOOK AS B, STORES AS S

WHERE B.Isbn = S.B\_isbn AND S.Quantity < 5;

* 1. Give all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased

SELECT C.Username, B.Title

## FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O, BOOK AS B, AUTHOR AS A, WRITES AS W

WHERE C.Username=O.Customer\_username AND O.Order\_num=AT.O\_order\_num AND B.Isbn=AT.B\_isbn

AND B.Isbn=W.B\_isbn AND W.A\_id=A.Id

AND A.Lname = 'Pratchett';

* 1. Find the total number of books purchased by a single customer (you choose how to designate the customer)

SELECT SUM(AT.Quantity)

## FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O

WHERE C.Username=O.Customer\_username AND O.Order\_num=AT.O\_order\_num AND C.Username = 'iVan2'

GROUP BY O.Order\_num;

* 1. Find the customer who has purchased the most books and the total number of books they have purchased

SELECT Username, SUM(AT.Quantity) FROMCUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O

WHERE C.Username=O.Customer\_username AND O.Order\_num=AT.O\_order\_num

GROUP BY O.Order\_num HAVING SUM(AT.Quantity) IN (

SELECT MAX(Total\_book)

FROM (SELECT SUM(AT.Quantity) AS Total\_book

## FROM CUSTOMER AS C, ADDED\_TO AS AT, ORDERS AS O

WHERE C.Username=O.Customer\_username AND O.Order\_num=AT.O\_order\_num

GROUP BY O.Order\_num)

);

1. For Project Checkpoint 02, you were asked to come up with three additional interesting queries that your database can provide. Give what those queries are supposed to retrieve in plain English, as relational algebra and then as SQL. Your queries should include joins and at least one should include an aggregate function, and they should be the same as the queries you outlined for Worksheet 02. If you were instructed to fix the queries in Checkpoint 02, make sure you use the fixed queries here. These queries should be provided in a plain text file named “WorksheetTwoExtraQueries.txt”.
2. Retrieve the usernames of customers who use Paypal for payment

𝜋 Username (CUSTOMER ⋈ Username = Customer\_username (ORDER ⋈ Paypal\_username = Username PAYPAL)) SELECT C.Username

## FROM CUSTOMER AS C, ORDERS AS O

WHERE C.Username = O.Customer\_username AND O.Paypal\_username IS NULL;

1. Find the warehouse with the greatest quantity of a certain book (specified by ISBN = 9783161484100)

𝜋 Id, Location, Max\_Quantity (𝑭 MAX Quantity (WAREHOUSE ⋈ Id = W\_id (𝜎 B\_isbn = ‘9783161484100’ STORES))) SELECT W.Id, W.Location

## FROM STORES AS S, WAREHOUSE AS W

WHERE W.Id = S.W\_id AND S.B\_isbn = 9783161484100

GROUP BY S.B\_isbn

HAVING S.Quantity = MAX(S.Quantity);

1. Find all the items to refunded on orders from a specific date (2022-10-10)

𝜋 Item (REFUND ⋈ Ordernum=Ordernum (𝜎 Date = ‘2022-10-10’ ORDER))

SELECT DISTINCT R.Item AS refunded\_book\_isbn FROM REFUND AS R, ORDERS AS O

WHERE R.Order\_num = O.Order\_num AND O.Order\_date = '2022-10-10';

1. Given your relational schema, provide the SQL for the following more advanced queries. These queries may require you to use techniques such as nesting, aggregation using having clauses, and other techniques . If your database schema does not contain the information to answer to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. **Note that if your database does contain the information but in non-aggregated form, you should NOT revise your model but instead figure out how to aggregate it for the query!** These queries should be provided in a plain text file named

“WorksheetTwoAdvancedQueries.txt”.

* 1. Provide a list of customer names, along with the total dollar amount each customer has spent. SELECT C.Username, SUM(AT.Quantity\*B.Price)

## FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

WHERE C.Username = O.Customer\_username AND O.Order\_num = AT.O\_order\_num AND AT.B\_isbn = B.Isbn

GROUP BY C.Username;

* 1. Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.
     + SELECT T.Name, T.Email, T.Each\_total
     + FROM (SELECT C.Username AS Name, C.Email AS Email, SUM(AT.Quantity\*B.Price) AS Each\_total

## FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

* + - WHERE C.Username = O.Customer\_username
    - AND O.Order\_num = AT.O\_order\_num
    - AND AT.B\_isbn = B.Isbn
    - GROUP BY C.Username) AS T
    - GROUP BY T.Name
    - HAVING T.Each\_total > AVG((SELECT SUM(AT.Quantity\*B.Price)

## FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

* + - WHERE C.Username = O.Customer\_username
    - AND O.Order\_num = AT.O\_order\_num
    - AND AT.B\_isbn = B.Isbn
    - GROUP BY C.Username));
  1. Provide a list of the titles in the database and associated total copies sold to customers, sorted from the title that has sold the most individual copies to the title that has sold the least.
     + SELECT B.Title, SUM(AT.Quantity) AS total\_copies

## FROM BOOK AS B, ADDED\_TO AS AT

* + - WHERE B.Isbn = AT.B\_isbn
    - GROUP BY B.Title
    - ORDER BY total\_copies DESC;
  1. Provide a list of the titles in the database and associated dollar totals for copies sold to customers, sorted from the title that has sold the highest dollar amount to the title that has sold the smallest.
     + SELECT B.Title, SUM(AT.Quantity)\*B.Price AS dollar\_totals

## FROM BOOK AS B, ADDED\_TO AS AT

* + - WHERE B.Isbn = AT.B\_isbn
    - GROUP BY B.Title
    - ORDER BY dollar\_totals DESC;
  1. Find the most popular author in the database (i.e. the one who has sold the most books)

## FROM BOOK AS B, AUTHOR AS A, WRITES AS W

* + - WHERE B.Isbn=W.B\_isbn
    - AND A.Id=W.A\_Id
    - AND B.Isbn IN (SELECT AT.B\_isbn

## FROM ADDED\_TO AS AT

* + - GROUP BY AT.B\_isbn
    - HAVING SUM(AT.Quantity) >= (SELECT MAX(Summation.S)
    - FROM (SELECT SUM(Quantity) AS S FROM ADDED\_TO GROUP BY B\_isbn) AS

Summation)

- );

* 1. Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)
     + SELECT A.FName, A.LName

## FROM BOOK AS B, AUTHOR AS A, WRITES AS W

* + - WHERE B.Isbn=W.B\_isbn
    - AND A.Id=W.A\_Id
    - AND B.Isbn IN (SELECT AT.B\_isbn

## FROM ADDED\_TO AS AT, BOOK AS B

* + - WHERE B.Isbn=AT.B\_isbn
    - GROUP BY AT.B\_isbn
    - HAVING SUM(AT.Quantity)\*B.Price >= (SELECT MAX(Summation.D)
    - FROM (SELECT SUM(AT.Quantity)\*B.Price AS D FROM ADDED\_TO AS AT,

BOOK AS B WHERE B.Isbn=AT.B\_isbn GROUP BY AT.B\_isbn) AS Summation)

- );

* 1. Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.
     + CREATE VIEW Four\_g AS
     + SELECT O.Customer\_username

## FROM ORDERS AS O, BOOK AS B, WRITES AS W, ADDED\_TO AS AT

* + - WHERE O.Order\_num=AT.O\_order\_num AND B.Isbn=AT.B\_isbn AND W.B\_isbn=B.Isbn AND
    - W.A\_id IN (
    - SELECT A.Id

## FROM BOOK AS B, AUTHOR AS A, WRITES AS W

* + - WHERE B.Isbn=W.B\_isbn
    - AND A.Id=W.A\_Id
    - AND B.Isbn IN (SELECT AT.B\_isbn

## FROM ADDED\_TO AS AT, BOOK AS B

* + - WHERE B.Isbn=AT.B\_isbn
    - GROUP BY AT.B\_isbn
    - HAVING SUM(AT.Quantity)\*B.Price >= (SELECT MAX(Summation.D)
    - FROM (SELECT SUM(AT.Quantity)\*B.Price AS D FROM ADDED\_TO AS AT,

BOOK AS B WHERE B.Isbn=AT.B\_isbn GROUP BY AT.B\_isbn) AS Summation)

- ));

* 1. Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.
     + SELECT DISTINCT A.Fname, A.Lname

## FROM ADDED\_TO AS AT, BOOK AS B, WRITES AS W, AUTHOR AS A

* + - WHERE AT.B\_isbn=B.Isbn AND B.Isbn=W.B\_isbn AND W.A\_id=A.Id AND
    - AT.O\_order\_num IN (
    - SELECT T.Onum
    - FROM (SELECT C.Username AS Name, O.Order\_num AS Onum, SUM(AT.Quantity\*B.Price) AS Each\_total

## FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

* + - WHERE C.Username = O.Customer\_username
    - AND O.Order\_num = AT.O\_order\_num
    - AND AT.B\_isbn = B.Isbn
    - GROUP BY C.Username) AS T
    - GROUP BY T.Name
    - HAVING T.Each\_total > AVG((SELECT SUM(AT.Quantity\*B.Price)

## FROM CUSTOMER AS C, ORDERS AS O, ADDED\_TO AS AT, BOOK AS B

* + - WHERE C.Username = O.Customer\_username
    - AND O.Order\_num = AT.O\_order\_num
    - AND AT.B\_isbn = B.Isbn
    - GROUP BY C.Username)));

Once you have completed all of the questions for Part Two, create a ZIP archive containing the binary SQLite file and the three text files and submit this to the Carmen Dropbox. **Make sure your queries work against your database and provide your expected output before you submit them!**

**Feedback**

Sql works ok, but the data is all wrong! You need to implement the books from the data provided in the Project page on Carmen.

**See Part 2 BookStore database file for revised version – including all data provided in the Project page on Carmen**

**See Section 2 Page 7-10 for the new relational algebra for above queries**