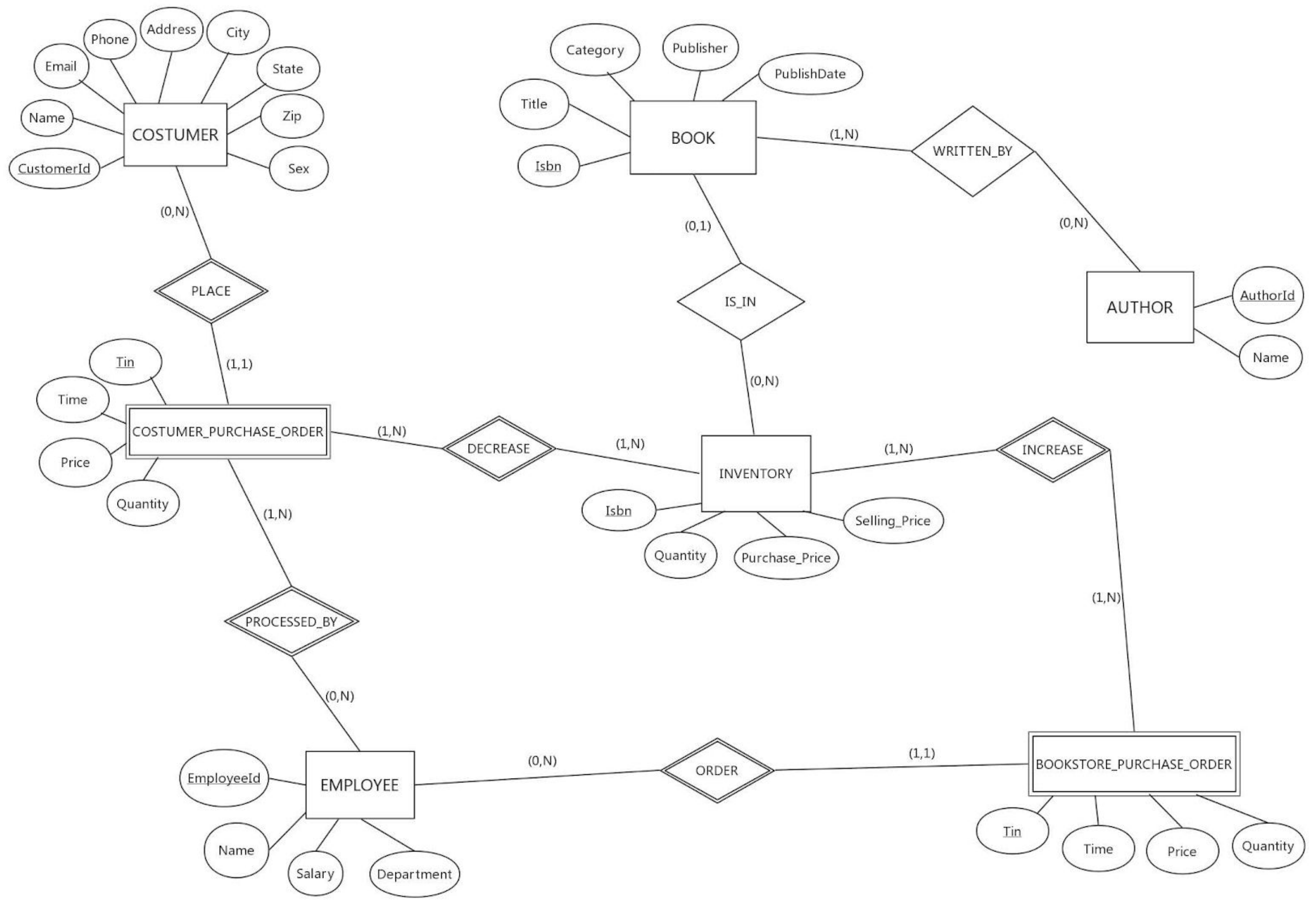
CSE 3241 Project Checkpoint 03 – SQL and More SQL

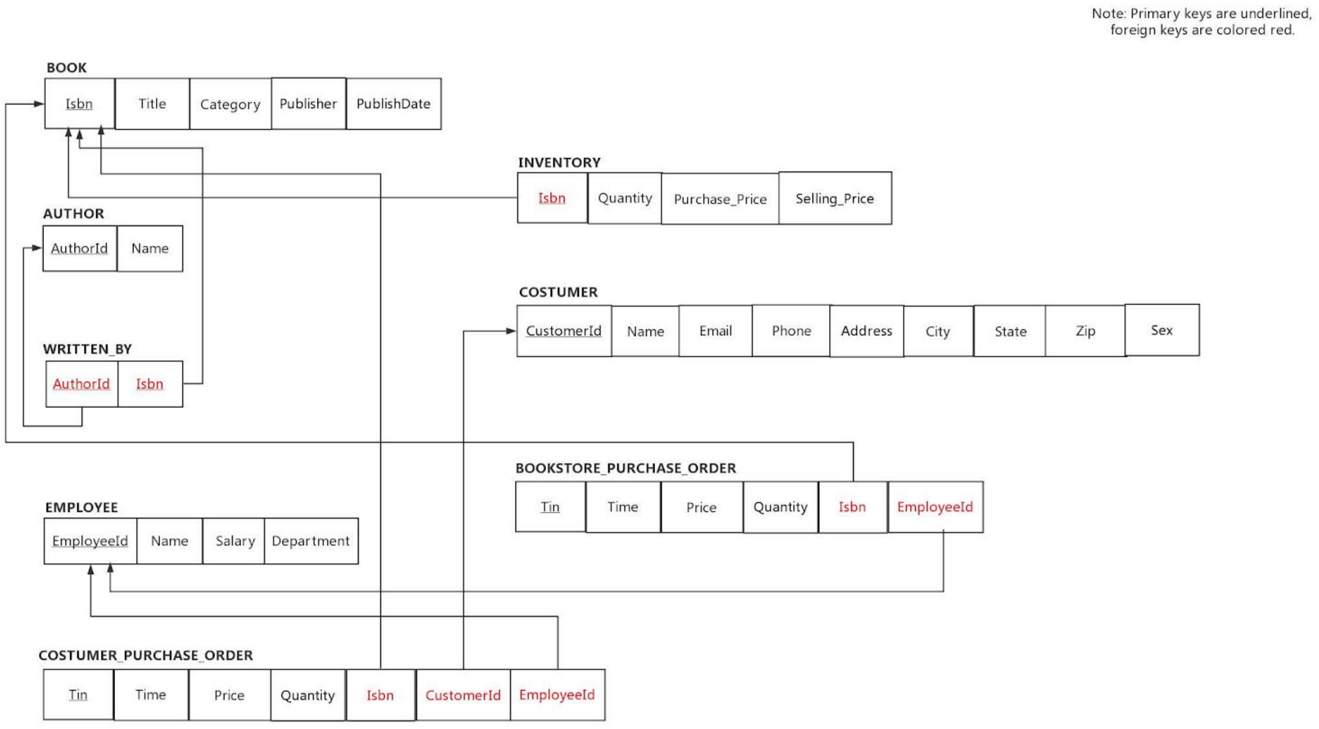
Names	Cenglin Bao, Wen Fang, Patrick Flanagan, Kai Li	Date 3/13/2020

This worksheet comes in two parts. The first part must be handed in in-class on the due date. The second part must be submitted to the Carmen Dropbox as a ZIP archive (and ONLY as a ZIP archive – no RAR or tar or ARC archives).

Part One – Due in class:

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.**





Part Two – Submitted to the Carmen Dropbox

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":
   a. Find the titles of all books by Pratchett that cost less than $10

SELECT B.TITLE
FROM BOOK AS B, AUTHOR AS A, WRITTEN_BY AS W, INVENTORY AS I
WHERE A.AUTHORID = W.AUTHORID AND B.ISBN = W.ISBN AND I.ISBN = B.ISBN AND A.NAME LIKE '% PRATCHETT' AND I.PURCHASE_PRICE < 10;

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

SELECT B.TITLE, CPO.TIME
FROM BOOK AS B, CUSTOMER_PURCHASE_ORDER AS CPO, CUSTOMER AS C
WHERE C.CUSTOMERID = CPO.CUSTOMERID AND CPO.ISBN = B.ISBN AND C.NAME = 'ISAAC NEWTON';

   c. Find the titles and ISBNs for all books with less than 5 copies in stock

SELECT B.TITLE, B.ISBN
FROM BOOK AS B, INVENTORY AS I

```
WHERE B.ISBN = I.ISBN AND I.QUANTITY < 5;
```

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

```
SELECT C.NAME, B.TITLE
FROM CUSTOMER AS C, BOOK AS B, CUSTOMER_PURCHASE_ORDER AS CPO, AUTHOR AS A, WRITTEN_BY AS W
WHERE C.CUSTOMERID = CPO.CUSTOMERID AND B.ISBN = CPO.ISBN AND B.ISBN = W.ISBN AND A.AUTHORID =
W.AUTHORID AND A.NAME LIKE '% PRATCHETT';
```

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

```
SELECT COUNT(*) AS BOOKS_PURCHASED
FROM CUSTOMER AS C, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE CPO.CUSTOMERID = C.CUSTOMERID AND C.NAME = 'ISAAC NEWTON';
```

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

```
SELECT C.NAME, MAX(BOOKS_PURCHASED)
FROM
(SELECT C.NAME, SUM(CPO.QUANTITY) AS BOOKS_PURCHASED
FROM CUSTOMER AS C, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE C.CUSTOMERID = CPO.CUSTOMERID
GROUP BY C.CUSTOMERID);
```

3. 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”.

4. 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”.

a. Provide a list of customer names, along with the total dollar amount each customer has spent.

```
SELECT C.NAME, DOLLARS_SPENT
FROM
(SELECT C.NAME, SUM(CPO.PRICE * CPO.QUANTITY) AS DOLLARS_SPENT
FROM CUSTOMER AS C, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE C.CUSTOMERID = CPO.CUSTOMERID
GROUP BY C.CUSTOMERID);
```

b. Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.

```
SELECT C.NAME, C.EMAIL
FROM CUSTOMER AS C, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE C.CUSTOMERID = CPO.CUSTOMERID
GROUP BY C.CUSTOMERID
HAVING SUM(CPO.PRICE * CPO.QUANTITY) >
(SELECT AVG(DOLLARS_SPENT)
FROM
(SELECT SUM(CPO.PRICE * CPO.QUANTITY) AS DOLLARS_SPENT
FROM CUSTOMER AS C, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE C.CUSTOMERID = CPO.CUSTOMERID
GROUP BY C.CUSTOMERID));
```

c. 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(CPO.QUANTITY) AS TOTAL_SOLD
FROM BOOK AS B, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE B.ISBN = CPO.ISBN
GROUP BY B.ISBN
ORDER BY TOTAL_SOLD DESC;
```

d. 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(CPO.QUANTITY * CPO.PRICE) AS TOTAL_SALES
FROM BOOK AS B, CUSTOMER_PURCHASE_ORDER AS CPO
WHERE B.ISBN = CPO.ISBN
GROUP BY B.ISBN
ORDER BY TOTAL_SALES DESC;
```

e. Find the most popular author in the database (i.e. the one who has sold the most books)

```
SELECT A.NAME, MAX(TOTAL_BOOKS_SOLD)
FROM
(SELECT A.NAME, SUM(CPO.QUANTITY) AS TOTAL_BOOKS_SOLD
FROM AUTHOR AS A, CUSTOMER_PURCHASE_ORDER AS CPO, WRITTEN_BY AS W
WHERE A.AUTHORID = W.AUTHORID AND W.ISBN = CPO.ISBN
GROUP BY A.AUTHORID);
```

f. Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)

```
SELECT A.NAME, MAX(TOTAL_SALES)
FROM
(SELECT A.NAME, SUM(CPO.QUANTITY * CPO.PRICE) AS TOTAL_SALES
FROM AUTHOR AS A, CUSTOMER_PURCHASE_ORDER AS CPO, WRITTEN_BY AS W
WHERE A.AUTHORID = W.AUTHORID AND W.ISBN = CPO.ISBN
GROUP BY A.AUTHORID);
```

g. Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.

```
SELECT C.*
FROM CUSTOMER AS C, WRITTEN_BY AS W, CUSTOMER_PURCHASE_ORDER AS CPO,
(SELECT A.AUTHORID
FROM
(
SELECT A.AUTHORID, MAX(TOTAL_SALES)
FROM
(SELECT A.AUTHORID, SUM(CPO.QUANTITY * CPO.PRICE) AS TOTAL_SALES
FROM AUTHOR AS A, CUSTOMER_PURCHASE_ORDER AS CPO, WRITTEN_BY AS W
WHERE A.AUTHORID = W.AUTHORID AND W.ISBN = CPO.ISBN
GROUP BY A.AUTHORID) AS A) AS A) AS X
WHERE X.AUTHORID = W.AUTHORID AND W.ISBN = CPO.ISBN AND C.CUSTOMERID = CPO.CUSTOMERID;
```

h. Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```
CREATE VIEW AVG_SPEND
AS SELECT avg(PRICE*QUANTITY) AS AVG
FROM CUSTOMER_PURCHASE_ORDER

CREATE VIEW SELECTED_ORDER
AS SELECT CUSTOMERID, (PRICE*QUANTITY) AS SPEND, ISBN
FROM CUSTOMER_PURCHASE_ORDER

CREATE VIEW SELECTED_ISBN
AS SELECT O.ISBN
FROM SELECTED_ORDER AS O, AVG_SPEND AS A
GROUP BY O.ISBN
HAVING O.SPEND > A.AVG

SELECT A.NAME
FROM AUTHOR AS A, WRITTEN_BY AS W, SELECTED_ISBN AS S
WHERE A.AUTHORID = W.AUTHORID AND W.ISBN = S.ISBN
```

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!**