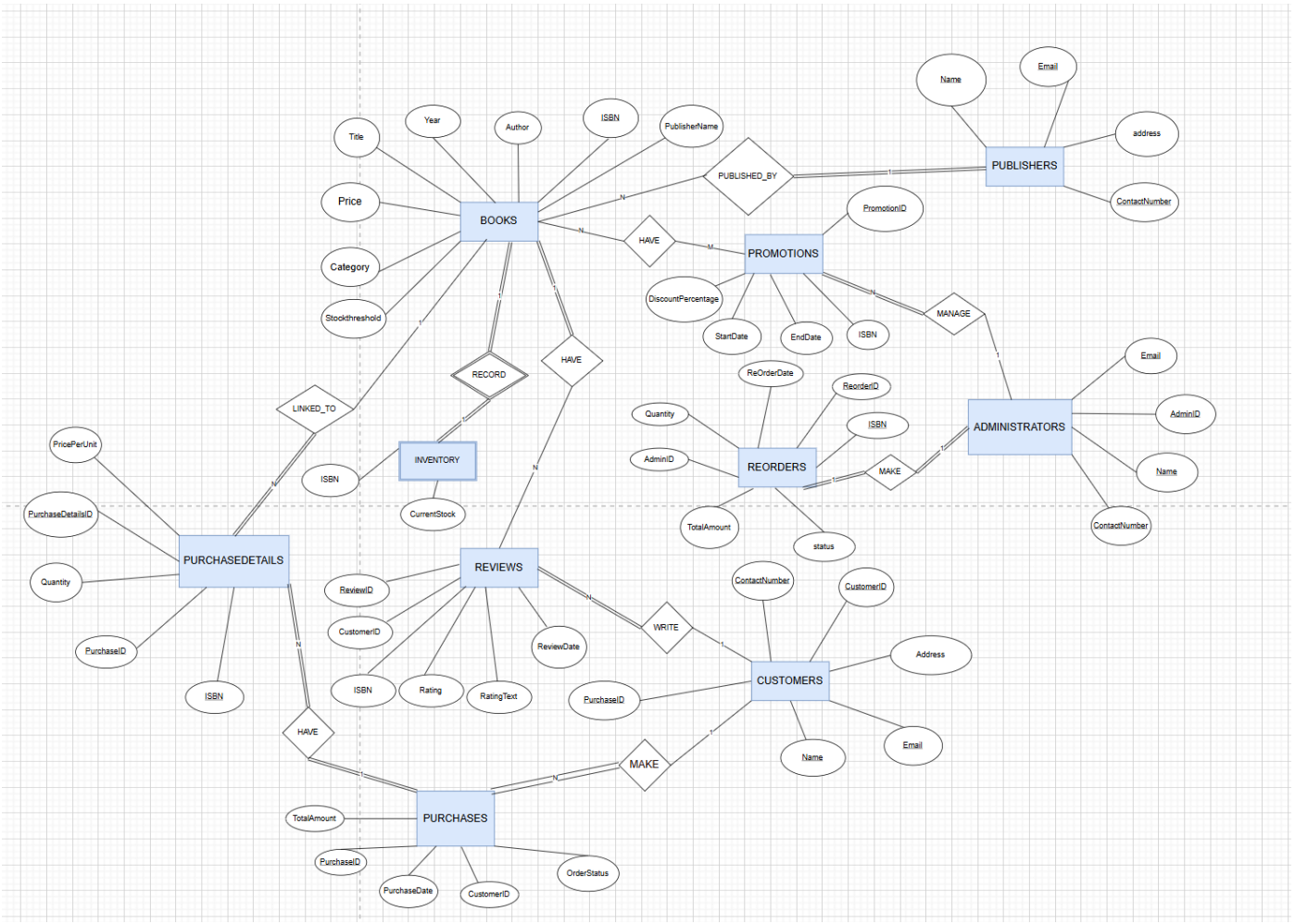
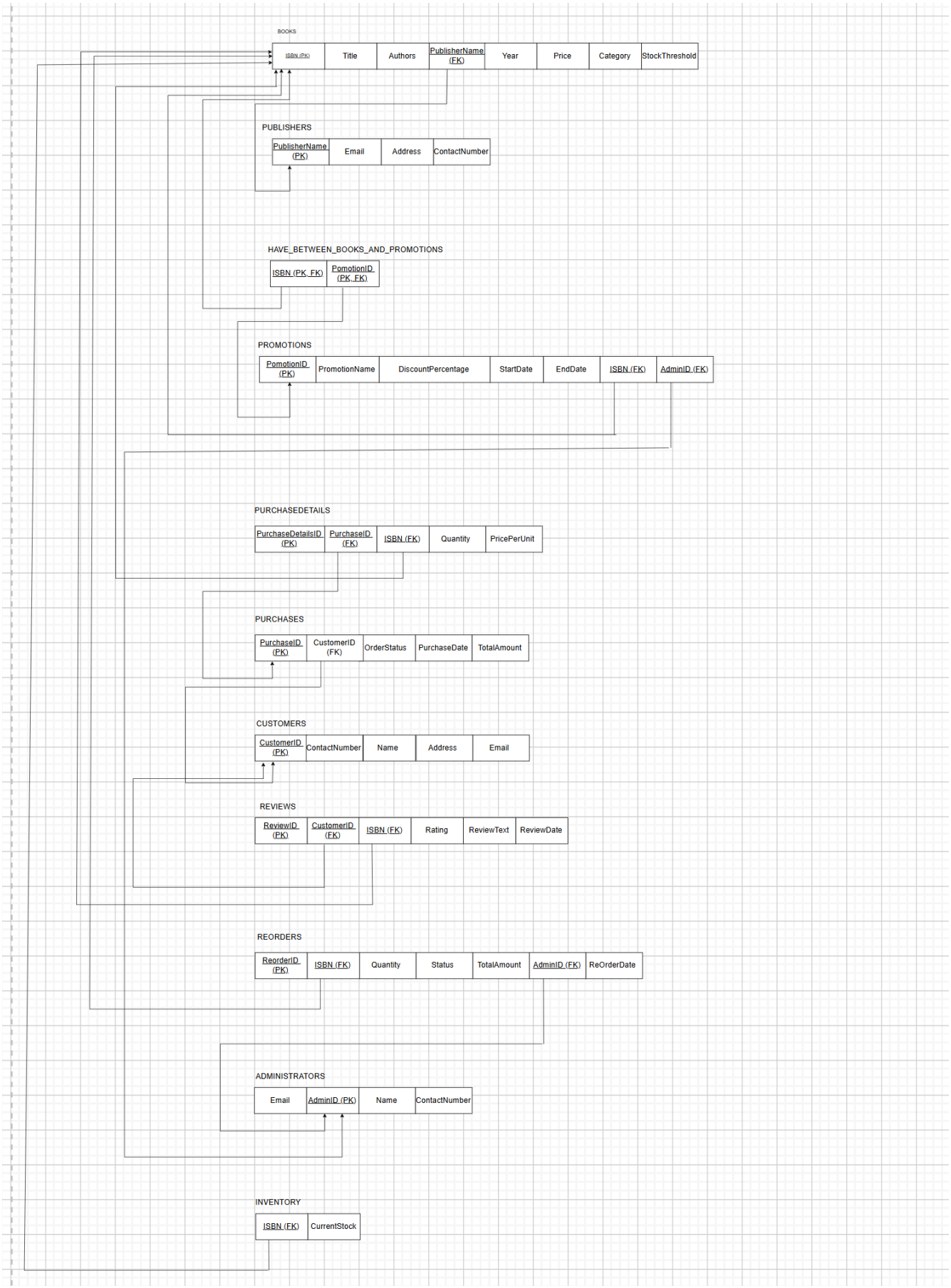
CSE 3241 Project Checkpoint 03 – SQL and More SQL

Names	Date
Yujie Yang, Yuang Li, zhengyang peng	

Submitted to the Carmen Dropbox

1. 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**

2. 


BOOKS
ISBN (PK)
Title
Authors
PublisherName (FK)
Year
Price
Category
StockThreshold
PUBLISHERS
PublisherName (PK)
Email
Address
ContactNumber
HAVE_BETWEEN_BOOKS_AND_PROMOTIONS
ISBN (PK, FK)
PomotionID (PK, FK)
PROMOTIONS
PomotionID (PK)
PromotionName
DiscountPercentage
StartDate
EndDate
ISBN (FK)
AdminID (FK)
PURCHASEDETAILS
PurchaseDetailsID (PK)
PurchaseID (FK)
ISBN (FK)
Quantity
PricePerUnit
PURCHASES
PurchaseID (PK)
CustomerID (FK)
OrderStatus
PurchaseDate
TotalAmount
CUSTOMERS
CustomerID (PK)
ContactNumber
Name
Address
Email
REVIEWS
ReviewID (PK)
CustomerID (FK)
ISBN (FK)
Rating
ReviewText
ReviewDate
REORDERS
ReorderID (PK)
ISBN (FK)
Quantity
Status
TotalAmount
AdminID (FK)
ReOrderDate
ADMINISTRATORS
Email
AdminID (PK)
Name
ContactNumber
INVENTORY
ISBN (FK)
CurrentStock

3. 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.

```
CREATE TABLE BOOKS (
    ISBN VARCHAR(13) PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Authors VARCHAR(255) NOT NULL,
    PublisherName VARCHAR(100),
    Year INT,
    Price DECIMAL(10, 2), -- Price with 2 decimal places (e.g., 29.99)
    Category VARCHAR(50),
    StockThreshold INT,
    FOREIGN KEY (PublisherName) REFERENCES PUBLISHERS(PublisherName)
);
CREATE TABLE PUBLISHERS (
    PublisherName VARCHAR(100) PRIMARY KEY,
    Email VARCHAR(100),
    Address VARCHAR(255),
    ContactNumber VARCHAR(20)
);
CREATE TABLE PROMOTIONS (
    PromotionID INT PRIMARY KEY,
    PromotionName VARCHAR(100) NOT NULL,
    DiscountPercentage DECIMAL(5, 2) NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    ISBN VARCHAR(13),
    AdminID INT,
    FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN),
    FOREIGN KEY (AdminID) REFERENCES ADMINISTRATORS(AdminID)
);

CREATE TABLE HAVE_BETWEEN_BOOKS_AND_PROMOTION (
    ISBN VARCHAR(13),
    PromotionID INT,
    PRIMARY KEY (ISBN, PromotionID),
    FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN),
    FOREIGN KEY (PromotionID) REFERENCES PROMOTIONS(PromotionID)
);
CREATE TABLE PURCHASEDETAILS (
    PurchaseDetailsID INT PRIMARY KEY,
    PurchaseID INT,
```

```
    ISBN VARCHAR(13),
    Quantity INT NOT NULL,
    PricePerUnit DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (PurchaseID) REFERENCES PURCHASES(PurchaseID),
    FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN)
);
CREATE TABLE PURCHASES (
    PurchaseID INT PRIMARY KEY,
    CustomerID INT,
    OrderStatus VARCHAR(50) NOT NULL,
    PurchaseDate DATE NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
);
CREATE TABLE CUSTOMERS (
    CustomerID INT PRIMARY KEY,
    ContactNumber VARCHAR(20),
    Name VARCHAR(100) NOT NULL,
    Address VARCHAR(255),
    Email VARCHAR(100) NOT NULL
);
CREATE TABLE REVIEWS (
    ReviewID INT PRIMARY KEY,
    CustomerID INT,
    ISBN VARCHAR(13),
    Rating INT NOT NULL CHECK (Rating >= 1 AND Rating <= 5),
    ReviewText TEXT,
    ReviewDate DATE NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID),
    FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN)
);
CREATE TABLE REORDERS (
    ReorderID INT PRIMARY KEY,
    ISBN VARCHAR(13),
    Quantity INT NOT NULL CHECK (Quantity > 0),
    Status VARCHAR(50) NOT NULL, -- Status of the reorder (e.g., "Pending",
"Completed")
    TotalAmount DECIMAL(10, 2) NOT NULL,
    AdminID INT,
    ReOrderDate DATE NOT NULL,
    FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN),
    FOREIGN KEY (AdminID) REFERENCES ADMINISTRATORS(AdminID)
);
CREATE TABLE ADMINISTRATORS (
```

```
    Email VARCHAR(100) NOT NULL,
    AdminID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    ContactNumber VARCHAR(20)
);
CREATE TABLE INVENTORY (
    ISBN VARCHAR(13) PRIMARY KEY,
    CurrentStock INT NOT NULL CHECK (CurrentStock >= 0),
    FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN)
);
```

4. 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 title
FROM BOOKS
WHERE Authors = 'Pratchett' AND 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 Title, PurchaseDate
FROM CUSTOMERS
JOIN PURCHASES P on CUSTOMERS.CustomerID = P.CustomerID
JOIN PURCHASEDETAILS P2 on P.PurchaseID = P2.PurchaseID
JOIN BOOKS B on P2.ISBN = B.ISBN
WHERE P.CustomerID = 1;
```

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

```
SELECT Title, BOOKS.ISBN
FROM BOOKS
JOIN INVENTORY I on BOOKS.ISBN = I.ISBN
WHERE CurrentStock < 5;
```

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

```
SELECT CUSTOMERS.Name, B.Title
FROM CUSTOMERS
JOIN PURCHASES P on CUSTOMERS.CustomerID = P.CustomerID
JOIN PURCHASEDETAILS P2 on P.PurchaseID = P2.PurchaseID
JOIN BOOKS B on P2.ISBN = B.ISBN
WHERE B.Authors = 'Pratchett';
```

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

```
SELECT SUM(Quantity), CUSTOMERS.Name
FROM CUSTOMERS
JOIN PURCHASES P on CUSTOMERS.CustomerID = P.CustomerID
JOIN PURCHASEDETAILS P2 on P.PurchaseID = P2.PurchaseID
WHERE CUSTOMERS.CustomerID = 1;
```

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

```
SELECT CUSTOMERS.Name, SUM(P2.Quantity) AS TotalBooks
FROM CUSTOMERS
JOIN PURCHASES P ON CUSTOMERS.CustomerID = P.CustomerID
JOIN PURCHASEDETAILS P2 ON P.PurchaseID = P2.PurchaseID
GROUP BY CUSTOMERS.CustomerID
ORDER BY TotalBooks DESC
LIMIT 1;
```

5. 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”.

Query 1 goals: Find all customers who gave a 5-star review for any book.

Relational algebra: π_Name (σ_Rating=5 (CUSTOMERS ⋈ REVIEWS))

SQL: SELECT DISTINCT C.Name

FROM CUSTOMERS C

JOIN REVIEWS R ON C.CustomerID = R.CustomerID

WHERE R.Rating = 5;

Query 2 goals: Find the total number of purchases made per customer.

Relational algebra: γ_CustomerID, COUNT(PurchaseID)→TotalPurchases (PURCHASES)

SQL: SELECT C.Name, COUNT(P.PurchaseID) AS TotalPurchases

FROM CUSTOMERS C

JOIN PURCHASES P ON C.CustomerID = P.CustomerID

GROUP BY C.CustomerID;

Query 3 goals: Get the average rating of each book that has at least 2 reviews.

Relational algebra: γ_ISBN, AVG(Rating) (σ_count(ReviewID) ≥ 2 (REVIEWS))

SQL: SELECT B.Title, AVG(R.Rating) AS AvgRating

FROM BOOKS B

JOIN REVIEWS R ON B.ISBN = R.ISBN

GROUP BY B.ISBN

HAVING COUNT(R.ReviewID) >= 2;

6.

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 clauses, and other techniques. If your database schema does not contain the information to answer 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, SUM(P.TotalAmount) AS TotalSpent
FROM CUSTOMERS C
JOIN PURCHASES P ON C.CustomerID = P.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 CUSTOMERS C
JOIN PURCHASES P ON C.CustomerID = P.CustomerID
GROUP BY C.CustomerID
HAVING SUM(P.TotalAmount) > (
  SELECT AVG(TotalSpent)
  FROM (
    SELECT SUM(TotalAmount) AS TotalSpent
    FROM PURCHASES
    GROUP BY 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(PD.Quantity) AS TotalSold
FROM BOOKS B
JOIN PURCHASEDETAILS PD ON B.ISBN = PD.ISBN
GROUP BY B.ISBN
ORDER BY TotalSold 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(PD.Quantity * PD.PricePerUnit) AS TotalRevenue
```

```
FROM BOOKS B
JOIN PURCHASEDETAILS PD ON B.ISBN = PD.ISBN
GROUP BY B.ISBN
ORDER BY TotalRevenue DESC;
```

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

```
SELECT B.Authors, SUM(PD.Quantity) AS TotalSold
FROM BOOKS B
JOIN PURCHASEDETAILS PD ON B.ISBN = PD.ISBN
GROUP BY B.Authors
ORDER BY TotalSold DESC
LIMIT 1;
```

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

```
SELECT B.Authors, SUM(PD.Quantity * PD.PricePerUnit) AS TotalProfit
FROM BOOKS B
JOIN PURCHASEDETAILS PD ON B.ISBN = PD.ISBN
GROUP BY B.Authors
ORDER BY TotalProfit DESC
LIMIT 1;
```

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

```
WITH MostProfitable AS (
  SELECT B.Authors
  FROM BOOKS B
  JOIN PURCHASEDETAILS PD ON B.ISBN = PD.ISBN
  GROUP BY B.Authors
  ORDER BY SUM(PD.Quantity * PD.PricePerUnit) DESC
  LIMIT 1
)
SELECT DISTINCT C.*
FROM CUSTOMERS C
JOIN PURCHASES P ON C.CustomerID = P.CustomerID
JOIN PURCHASEDETAILS PD ON P.PurchaseID = PD.PurchaseID
JOIN BOOKS B ON PD.ISBN = B.ISBN
JOIN MostProfitable M ON B.Authors = M.Authors;
```

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

```
WITH HighSpenders AS (
   SELECT CustomerID
   FROM PURCHASES
   GROUP BY CustomerID
   HAVING SUM(TotalAmount) > (
      SELECT AVG(TotalAmount)
      FROM PURCHASES
      GROUP BY CustomerID
   )
)
SELECT DISTINCT B.Authors
FROM BOOKS B
JOIN PURCHASEDETAILS PD ON B.ISBN = PD.ISBN
JOIN PURCHASES P ON PD.PurchaseID = P.PurchaseID
WHERE P.CustomerID IN (SELECT CustomerID FROM HighSpenders);
```

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!**