THE OHIO STATE UNIVERSITY

COLLEGE OF ENGINEERING

CSE 3241

Project : Bits & Books Final Report

29 December 2020

Our Team:

Alfath Ahmed

Akshath Srinivasan

Satwinder Singh

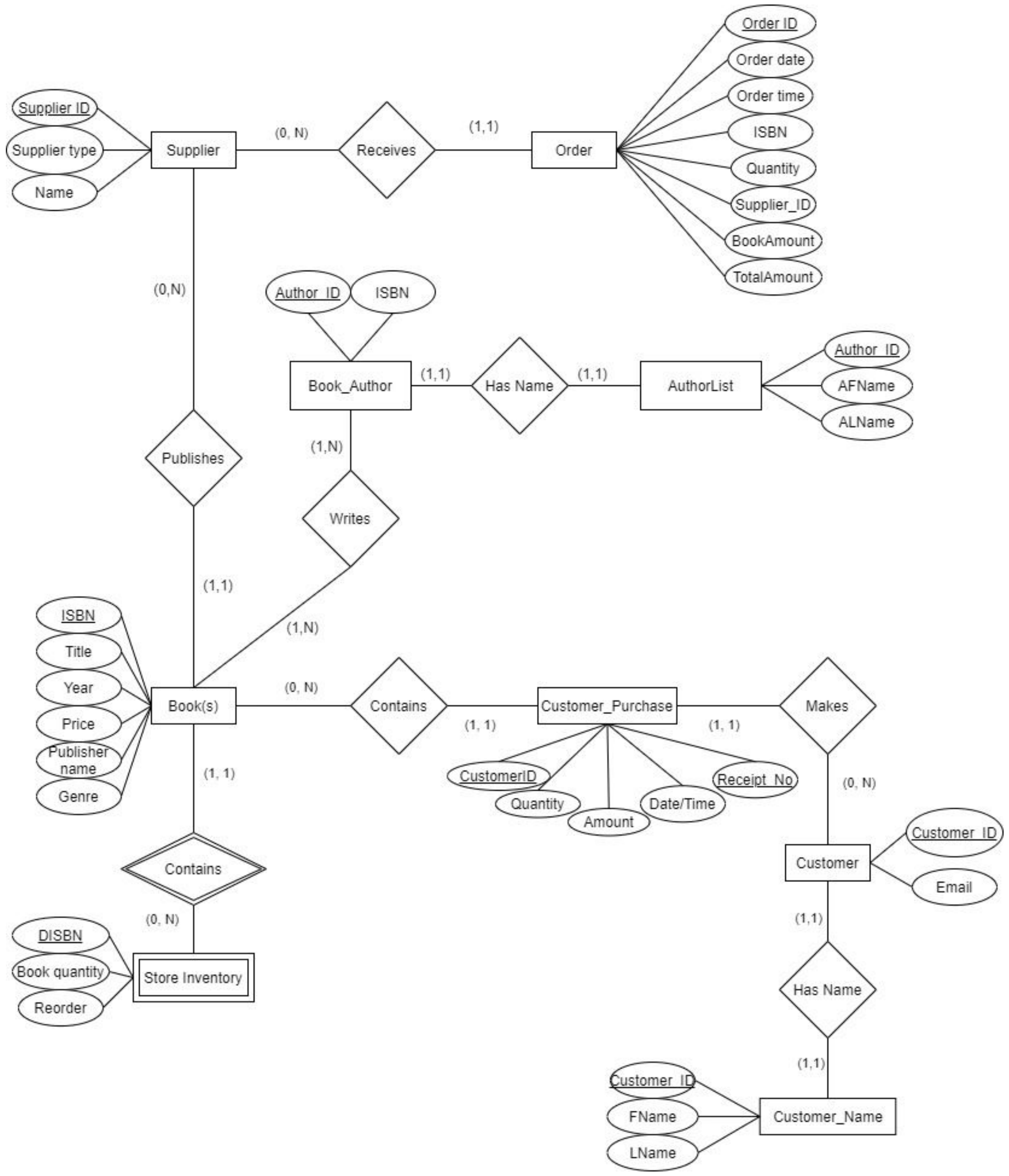
Mark Maher

Table of Contents:

Section 1 - Database Description

A database description document that contains the following information about your database (compiled from your completed and revised checkpoints):

a) A professionally presented, well-formatted ER-model that reflects the updates you have made during the semester. Do not submit a hand-drawn diagram.



b) A professionally presented, well-formatted relational schema for the database. This schema must be annotated with the primary key for each table, all foreign keys on all tables, and all functional dependencies on all tables. Make sure that connections between FKs and PKs are clear.
<u>NOTE:</u> primary keys are **<u>underlined</u>**, foreign keys are ***italicized***

- Supplier - (<u>Supplier_ID</u>, Supplier Type, Name)
- Order - (<u>Order_ID</u>, OrderDate, orderTime, *ISBN*, Quantity, *Supplier_ID,* BookAmount, TotalAmount)
- Book - (<u>ISBN</u>, title, year, price, publisher name, genre, *Supplier_ID*)
- Customer -(<u>Customer_ID</u>, Email)
- Customer_Purchase - (<u>Receipt_No</u>, <u>*ISBN*</u>, Title, Date_Time, Amount, *Customer_ID*, Quantity)
- Store_Inventory (<u>DISBN</u>, Book Quantity, ReOrder)
- Book_Author - (*ISBN*, <u>AuthorId</u>)
- AuthorList - (<u>Author_Id</u>, AFirst, ALast)
- Customer_Name - (<u>Customer_ID</u>, First, Last)

c) For each table, give a brief description of the level of normalization achieved for that table. If the table is not in BCNF, explain why.

Supplier : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Order : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Book : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Customer : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Customer_Purchase : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Store_Inventory : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Book_Author : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

AuthorList : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Customer_Name : This table is in Boyce Codd Normal Form & satisfies Third Normal Form conditions.

Every table is not only in third normal form, but is also in Boyce Codd normal form as well.

d) A description of each of the indexes that you have chosen to implement on your database, along with rationale for each.

Although the usage of a database index would allow us to efficiently retrieve data from a database. Our group decided to not implement indexes for simplicity of our database.

e) For each view that you have implemented, provide the following:

i. A brief description in English of what this view produces, and why it would be useful.
ii. Relational algebra expression to produce this view.
iii. SQL statements to produce the view.
iv. Sample output from the view, with 5-10 lines of data records shown.

Total Customer Bill

- The following view explains the total customer bill or how much money each customer has spent in the store.

```
CREATE VIEW TotalCustomerBill AS
        SELECT  N.Customer_ID, N.First, N.Last, SUM(CP.Amount)
        FROM Customer_Name N, Customer C, Customer_Purchase CP
        WHERE N.Customer_ID = C.Customer_ID AND C.Customer_ID = CP.Customer_ID
        GROUP BY N.Customer_ID;
```

Total Books Sold

- The following view shows the user how many of each book has been sold. This helps gauge the popularity of each book.

```
CREATE VIEW TotalBooksSold AS
        SELECT  B.ISBN, B.title, B.publisher_name, B.year, SUM(CP.Quantity)
        FROM Book B, Customer_Purchase CP
        WHERE B.ISBN = CP.ISBN
        GROUP BY B.ISBN;
```

f) A professionally presented description of three sample transactions useful for your database. This should include the sample SQL code for each transaction as well as an English language description of what "unit of work" the transaction represents. Remember – a transaction is a sequence of SQL statements taken as a unit – this can be reads and writes together or just a sequence of writes. One example of a sample transaction you might want to consider is the user making changes to an order – what might need to be considered a transaction in that case?

- The bookstore places an order of 200 copies of a book with the ISBN = 787960756 from Supplier with Supplier_ID = 1, so an order needs to be inserted into Orders.

```
BEGIN TRANSACTION NEW_Order;
        INSERT INTO Orders VALUES (21, '11/30/2020', '11:00', '787960756', 200, 1, 13.20, 0);
END TRANSACTION;
```

- The order of 200 copies of a book with the ISBN = 72227885 gets fulfilled, so the bookstore has 200 more copies of the book in the inventory.

```
BEGIN TRANSACTION Update_inventory;
        UPDATE Store_Inventory
        SET Book_Quantity = Book_Quantity + 200
        WHERE DISBN = '72227885';
END TRANSACTION;
```

- An existing customer with the Customer_ID = 1 purchases 5 copies of a book with the ISBN = 72227885, so the bookstore has 5 less copies of the book in the inventory, and a new Customer_Purchase entry is inserted.

```
BEGIN TRANSACTION Books_Purchased;
        INSERT INTO Customer_Purchase VALUES (21, '72227885', '11/29/2020 11:00', 87.45, 5, 1);
        UPDATE Store_Inventory
        SET Book_Quantity = Book_Quantity - 5
        WHERE DISBN = '72227885';
```

END TRANSACTION;

Section 2 - User Manual

A user manual describing the usage of your database, for use by developers who are going to be writing code to use your database. Your manual should include:

a) For each table, explain what real world entity it represents. Provide a description of each attribute, including its data type and any constraints you have built-in.

- Supplier - (<u>Supplier_ID</u>, Supplier Type, Name)
    - Supplier represents the person or organization that is providing our bookstore with the books.
    - [int] The Supplier_ID is our primary key and a unique identifier that you can use to locate specific supplier information.
    - [varchar[]] Supplier Type categorizes our supplier into groups for reporting and sorting purposes.
    - [varchar[]]Name represents the name of the supplier
- Order - (<u>Order_ID</u>, OrderDate, orderTime, <u>*ISBN*</u>, Quantity, BookAmount, TotalAmount ShipmentStatus, <u>*Supplier_ID*</u>)
    - Order represents a request made by the bookstore to the supplier to purchase certain goods
    - [int] Order_ID is a number system that the supplier uses exclusively to keep track of orders. Each order receives its own order ID that is not replicated
    - [DATE] OrderDate represents the data that the order was placed on
    - [TIME] OrderTime represents the time that the order was placed.
    - [int] ISBN is a 10-digit long specific number to identify a certain book.
    - [int] Quantity represents the book quantity that was ordered
    - [double] BookAmount represents the price of each book
    - [double] TotalAmount represents the total bill for the order
    - [boolean] ShipmentStatus showed the delivery information status of whether the order was delivered or not
- Book - (<u>ISBN</u>, title, year, price, publisher name, genre, <u>*Supplier_ID*</u>)
    - Book: represents a unique book that can be sold by a supplier & ordered by the store or a customer
    - [int] ISBN: An integer representing each books unique identifying number.
    - [varchar[]] title: Each book has a variably long string of characters for its title.
    - [YEAR] year: The date that the book in question was published.
    - [double] price: A double representing the monetary cost of each book.
    - [varchar[]] publisher name: The variably long string of characters representing the publisher's name.
    - [varchar[]] genre: The variable long string of character representing the genre, or theme of the contents of the book.
- Customer -(<u>Customer_ID</u>, Email)
    - Customer: Represents a client to the bookstore who may choose to purchase book(s) from the book store.

    - [int] Customer_ID: An integer representing each customer's unique identifying number.
    - [varchar[]] Email: A variably long string of characters representing the customer's contact information (email address).
- Customer_Purchase - (<u>Receipt_No</u>, *<u>ISBN</u>*, Date_Time, Amount, *<u>Customer_ID</u>*, Quantity)
    - Customer_Purchase: Represents a transaction made between the customer & the book store in exchange for money & book(s).
    - [int] Receipt_No: A uniquely identifying integer for each customer_purchase.
    - [DATETIME] Date_Time: The date & time that the transaction between customer & bookstore took place.
    - [double] Amount: A double representing how much money was given to the bookstore in exchange for their requested books.
    - [int] Quantity: The number of books the customer purchased from the bookstore.
- Store_Inventory (<u>DISBN</u>, Book Quantity, ReOrder)
    - [int] DISBN: This represents dependant ISBN value which is used to identify a certain book
    - [int] Book Quantity: This attribute represents the number of books in stock.
    - [Boolean] ReOrder: This represents the status of whether we should place an order again or not
- Book_Author - (<u>ISBN</u>, <u>AuthorId</u>)
    - [int] ISBN: This represents ISBN value which is used to identify a certain book
    - [varchar[]] AuthorId: This represents the unique identification number of the author
- AuthorList - (<u>Author_Id</u>, First, Last)
    - [varchar[]] AuthorId: This represents the unique identification number of the author
    - [varchar[]] First: This represents the first name of the author
    - [varchar[]] Last: This represents the Last name of the author
- Customer_Name - (<u>Customer_ID</u>, FName, LName)
    - [varchar[]] Customer_Id: This represents the unique identification number of the customer
    - [varchar[]] FirstName: This represents the first name of the customer
    - [varchar[]] LastName: This represents the Last name of the customer

b) The sample SQL queries that you provided in Checkpoints 03 and 02. These queries should be organized and presented neatly and professionally. Each query should include:
   i. An English language description of what the query should be returning
   ii. The correct relational algebra syntax of the query
   iii. The equivalent SQL query

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

$$\Pi_{<\text{Title}>}(\boldsymbol{\sigma}_{<\text{Price} < 10 \text{ AND Author} = \text{'Pratchett'}>}(((\text{Book}) \bowtie_{<\text{ISBN} = \text{ISBN}>} (\text{Book\_Author})) \bowtie_{<\text{Author\_ID} = \text{Author\_ID}>} (\text{AuthorList})))$$

```
SELECT DISTINCT B.title, AL.AFirst, AL.ALast, B.price
FROM Book B, Book_Author BA, AuthorList AL
WHERE B.ISBN = BA.ISBN AND AL.AuthorId = BA.AuthorId AND AL.ALast = "Pratchett" AND B.price <
10.00;
```

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

$\Pi_{\text{<Title, Date/Time>}}(\sigma_{\text{<Fname = 'Billy' AND Lname = 'Bob' >}}((\text{Book}) \bowtie_{\text{<ISBN = ISBN>}} (\text{Customer\_Purchase})))$

(Assuming: Customer_ID = 1)

```
SELECT B.title, CP.Date_Time
FROM Customer_Purchase CP, Book B
WHERE Customer_ID = 1 AND B.ISBN = CP.ISBN;
```

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

$\Pi_{\text{<ISBN, Title>}}(\sigma_{\text{<Book\_Quantity < 5 >}}((\text{Store\_inventory}) \bowtie_{\text{<ISBN = ISBN>}} (\text{Book})))$

```
SELECT B.title, B.ISBN
FROM Store_Inventory SI, Book B
WHERE SI.Book_Quantity < 5 AND SI.DISBN = B.ISBN;
```

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

$\Pi_{\text{<Fname, Lname, Title>}}(\sigma_{\text{<ALName = 'Pratchett'>}}((((((\text{Book}) \bowtie_{\text{<ISBN = ISBN>}} (\text{Customer\_Purchase})) \bowtie_{\text{<Customer\_ID = Customer\_ID>}} (\text{Customer})) \bowtie_{\text{<Customer\_ID = Customer\_ID>}} (\text{Customer\_Name})) \bowtie_{\text{<ISBN = ISBN>}} (\text{Book\_Author})) \bowtie_{\text{<Author\_ID = Author\_ID>}} (\text{AuthorList})))$

```
SELECT CN.First,CN.Last, B.title
FROM Customer_Purchase CP, Book B, AuthorList A, Book_Author BA, Customer_Name CN, Customer C
WHERE C.Customer_ID = CP.Customer_ID AND C.Customer_ID = CN.Customer_ID AND CP.ISBN = B.ISBN
AND BA.ISBN = B.ISBN AND A.AuthorId = BA.AuthorId AND A.ALast = "Pratchett";
```

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

$F_{\text{COUNT}}(\sigma_{\text{<Customer\_ID = 'Customer\_ID'>}}(\text{Customer\_Purchase}))$

(Assuming: Customer_ID = 10)

```
SELECT SUM(CP.Quantity)
FROM Customer_Purchase CP
WHERE CP.Customer_ID = 10;
```

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

$F_{MAX}(_{<Customer\_ID>}F_{COUNT}(Customer\_Purchase) \bowtie_{<Customer\_ID = 'Customer\_ID'>}(Customer))$

```
SELECT CN.First,CN.Last, MAX(CP.Quantity) AS Total
FROM Customer_Purchase CP, Book B, AuthorList A, Book_Author BA, Customer_Name CN, Customer C
WHERE C.Customer_ID = CP.Customer_ID AND C.Customer_ID = CN.Customer_ID AND CP.ISBN = B.ISBN
AND BA.ISBN = B.ISBN AND A.AuthorId = BA.AuthorId;
```

Count the number of books by a particular author

$F_{COUNT}(\sigma_{<Author = 'Brutus''>}(Book)) \bowtie_{<ISBN>} Store\_Inventory)$

(Counting books by Pratchett)

```
SELECT SUM(S.Book_Quantity)
FROM Book_Author BA, Book B, AuthorList A, Store_Inventory S
WHERE BA.ISBN = B.ISBN AND S.DISBN = B.ISBN AND BA.AuthorId = A.AuthorId AND A.ALast =
"Pratchett";
```

Find the total number of customers who have purchased a certain book

$F_{COUNT}(\sigma_{<ISBN = ISBN'>}(Book)) \bowtie_{< Customer\_ID = 'Customer\_ID'>} Customer\_Purchase)$

(Assuming Title = ‘The Demon-Haunted World: Science As a Candle in the Dark’)

```
SELECT COUNT(*)
FROM Customer_Purchase CP, Book B
WHERE B.ISBN = CP.ISBN AND B.title = "The Demon-Haunted World: Science As a Candle in the Dark";
```

Find the title of the most purchased book by Carl Sagan

$_{<Title>}F_{MAX}(_{<ISBN, Title>}F_{COUNT(Receipt No.)}(Customer\_Purchase) \bowtie_{<Author= "Carl Sagan">}(Book))$

```
SELECT MAX(B.title)
FROM Customer_Purchase CP, Book_Author BA, Book B, AuthorList A
WHERE CP.ISBN = B.ISBN AND B.ISBN = BA.ISBN AND BA.AuthorId = A.AuthorId AND A.ALast = "Sagan";
```

c) INSERT syntax for adding new books, publishers, authors and customers to your system. If there are dependencies in your system that require multiple records to be added to tables in a specific order to add one of these items, make sure you clearly indicate what those restrictions are.

- Add a new book into the database

    - When we add a book into the database, we can notice that in our database entity we have the foreign key *supplier_ID* which refers to the supplier entity. So, if we want to add a book into the book entity, we have to first add a supplier into the supplier entity.

      insert syntax:

      ```
      insert into Supplier
      values("23", "Rockwell Publishers");

      insert into Books
      values("0123456789", "Brutus Horror Story", 2016, 10.00, "Rockwell Publishers", 23, "Horror");
      ```

- Add a publisher into the database
    - When we add a publisher into the database, we are assuming that the supplier is the publisher, so we can insert a new supplier into the supplier table.

      insert syntax:

      ```
      insert into Supplier
      values(24, "Brutus & Co.");
      ```

- Add a book author into the database along with their book
    - When we want to add an author to the database, we have to first insert the author into the AuthorList. Then we must add their book to Books, and link the two by inserting a tuple into Book_Author. This is because both Book_Author has foreign keys referring to the ISBN from Books and the Author_ID from AuthorList.

      insert syntax:

      ```
      insert into AuthorList
      values ("23", "Brutus", "Buckeye");

      insert into Books
      values("9876543210", "Bruticus the Great", 2020, 20.00, "Brutus & Company", 25, "Drama");

      insert into Book_Author
      values("9876543210", 25);
      ```

- Add a customer into the database
    - When we want to add a customer to our database, first we have to insert an entry into Customer, and then insert the Customer_Name entry because Customer_Name has a foreign key that refers to the Customer_ID in Customer.

      insert syntax:

      ```
      insert into Customer
      values(23, "brutus@osu.edu");

      insert into Customer
      values(23, "Brutus", "Buckeye");
      ```

d) DELETE syntax for removing books, publishers, authors and customers from your system. Again, indicate any dependencies that exist on the order that the steps in your DELETE must take. In addition, provide an example set of DELETE statements for each entity in your database.

- Delete a new book
  - When we want to delete a new book from our book, we need to first delete the table in which foreign keys of our table refer to our book entity table. We know that in our database, the Book_author entity has a foreign key ISBN referring to the book. So, we need to delete this table first then delete a book.

    Delete Syntax:

    ```
    delete from Book_Author
    where ISBN = 60502935;

    delete from Book
    where ISBN = 60502935;
    ```

- Delete a publisher in the database
  - In our database, we are assuming that the publisher is the supplier, so when we want to delete a publisher, we have to check whether there are any foreign keys pointing to Supplier_ID. Book and Orders refer to Supplier_ID, so we first need to delete the entries in Book and Orders where Supplier_ID = the one we want to delete.

    Delete Syntax:

    ```
    delete from Book
    where Supplier_ID = 1;

    delete from Orders
    where Supplier_ID = 1;

    delete from Supplier
    where Supplier_ID = 1;
    ```

- Delete an author in the database
  - When we want to delete an author, we notice that Book_Author has a foreign key referring to the AuthorList entity. So, we must first delete the entries from Book_Author, and then delete from AuthorList.

    Delete Syntax:

    ```
    delete from Book_Author
    where AuthorId = 1;

    delete from AuthorList
    where AuthorId = 1;
    ```

- Delete a customer in the database
  - When we want to delete a customer from our database, we can notice that Customer_ID is a foreign key in our database in the Customer_Purchase entity and the Customer_Name entity. So, we have to first

delete from customer_purchase entity then from the entity Customer_Name. Then we can delete the customer from Customer.

Delete Syntax:

```
delete from Customer_Purchase
where Customer_ID = 1;

delete from Customer_Name
where Customer_ID = 1;

delete from Customer
where Customer_ID = 1;
```

APPENDIX

No revision was required for any of the checkpoints except for Checkpoint 3. The 4 original checkpoints are pasted below. The final code, schema, and diagram can be seen above in the report.

CSE 3241 Project Checkpoint 01 – Entities and Relationships

In a **NEATLY TYPED** document, provide the following:

1. Based on the requirements given in the project overview, list the entities to be modeled in this database. For each entity, provide a list of associated attributes.
    - Supplier - Supplier ID, Location Address, Supplier Type, Name
    - Order - Transaction ID, OrderDate, orderTime, ISBN, shipmentID, Quantity
    - Transaction - Transaction ID, Client ID, Supplier ID, date, time, amount, tax, payer, type
    - Client - Client ID, Name, contact, balance, location
    - Book - ISBN, title, author, year, price, category, publisher name
    - Author - Name
    - Customer - Customer ID, Name, Payment info
    - Shipment - Supplier ID, Client ID, Shipment ID, Shipment Status, Content
    - Store Inventory - ISBN, Book Quantity, ReOrder

2. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, “CUSTOMER entities purchase BOOK entities”).
    - Client places order
    - Supplier receives order
    - Client makes payment transaction
    - Supplier receives transaction
    - Shipment is shipped by Supplier
    - Shipment is received by the store
    - Books are written by author
    - Books are published by publisher
    - Client sell book
    - Customer buys book
    - Inventory includes book

3. Propose at least two additional entities that it would be useful for this database to model beyond the scope of the project requirements. Provide a list of possible attributes for the additional entities and possible relationships they may have with each other and the rest of the entities in the database. Give a brief, one

sentence rationale for why adding these entities would be interesting/useful to the stakeholders for this database project.

a. Employee - Employee ID, Name, Role, Address, Schedule, Wage, Contact, Start date
    i. This will be used to keep track of the employees working at the stores.
b. Membership - Customer ID, Name, Address, Contact, Payment info
    i. This might be useful for customers who wish to make an account at the bookstore for offers and benefits.

4. Give at least four examples of some informal queries/reports that it might be useful for this database might be used to generate. Include one example for each of the additional entities you proposed in question 3 above.
    - Request a book's price
    - Assess how much stock is remaining
    - Request status of shipments (where they are, are they processed, etc…)
    - Filter books from a specific author
    - Filter books between a certain price ($15 < x < $45)
    - Filter books of a certain year
    - Filter books of a specific genre
    - Request a list of suppliers
    - Request a list of books from a specific publisher
    - Which employees work a certain day
    - List of employees by role
    - list of members of subscription plan

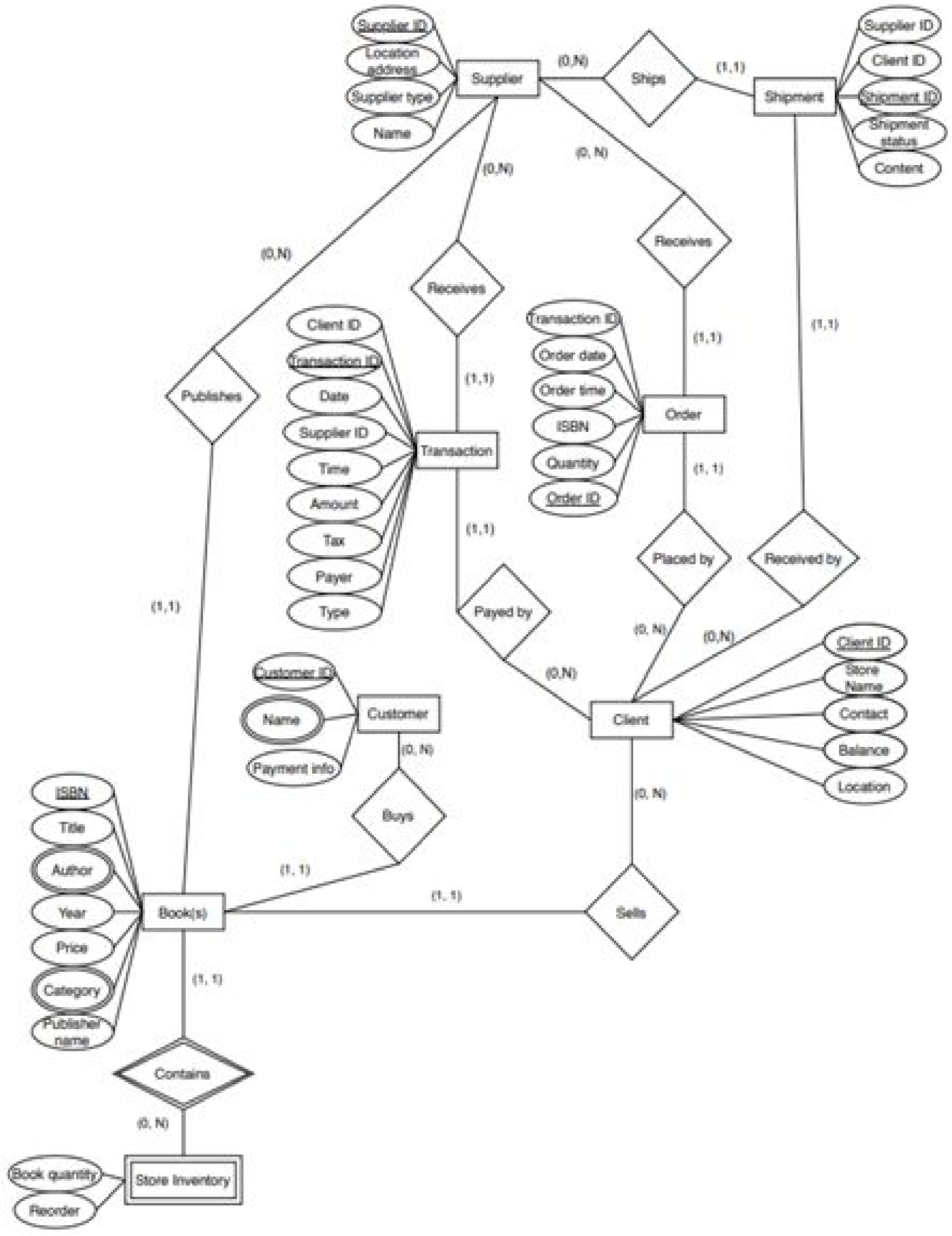
5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you've outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your model to allow for publishers to be added as separate entities.
    ● Name of Publisher
    ● Contact information of Publisher
    ● Location of Publisher
    - Yes, it is possible to add a new publisher because a publisher could be added as a Supplier entity without the need to link them to a Book.

6. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 3 above.

    A. Supplier no longer supplies the book
        a. Add an attribute under order "containsBook". This would be changed to false.
    B. Adding an audiobook/ ebook
        a. Have an attribute bookType under the book entity
    C. Employee quits job
        a. Have a boolean attribute "employed" under entity Employee

D. Customer subscribes to membership program
    a. Have a boolean attribute "Active" under entity Membership

7. Provide an ER diagram for your database. Make sure you include all of the entities and relationships you determined in the questions above ***INCLUDING the entities for question 3 above***, and remember that ***EVERY*** entity in your model needs to connect to another entity in the model via some kind of relationship.



CSE 3241 Project Checkpoint 02 – Relational Model and Relational Algebra

In a **NEATLY TYPED** document, provide the following:

1. Provide a current version of your ER Model as per Project Checkpoint 01. If you were instructed to change the model for Project Checkpoint 01, make sure you use the revised version of your ER Model.

    a. We were not advised to change anything on our ER diagrams

2. Map your ER model to a relational schema. Indicate all primary and foreign keys. Note Primary Keys are underlined and foreign keys are *underlined and italicized.*
    - Supplier - (Supplier ID, Location Address, Supplier Type, Name)
    - Order - (Order ID, OrderDate, orderTime, ISBN, Quantity, *Supplier ID*, *Client ID*)
    - Transaction - (Transaction ID, Client ID, Supplier ID, date, time, amount, tax, payer, type)
    - Client - (Client ID, Store Name, Contact, Balance, Location)
    - Book - (ISBN, title, year, price, publisher name, *Supplier ID*)
    - Customer -(Customer ID, Payment Info)
    - Customer_Purchase - (Receipt_No, ISBN, Title, Date/Time, Amount, *Customer ID*, *Client ID*)
    - Shipment - (Shipment ID, Supplier ID, Shipment Status, Content, *Supplier ID*, *Client ID*)
    - Store Inventory (DISBN, Book Quantity, ReOrder)
    - Buys (*Customer ID*, *ISBN*)
    - Sells (*Client ID*, *ISBN*)
    - Book_Author - (ISBN, author)
    - Category - (ISBN, category)
    - Customer_Name - (Customer ID, Customer Name)

3. Given your relational schema, provide the relational algebra 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:

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

    $\Pi_{\text{<Title>}}(\boldsymbol{\sigma}_{\text{<Price < 10 AND Author = 'Pratchett' >}}(\text{Book}))$

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

    1PersonPurchases <- SIGMA.CustomerID = 24 (Customer_Purchase)
    TitlesAndDates <- Pi.Title, Date/Time (1PersonPurchase))

    $\Pi_{\text{<Title, Date/Time>}}(\boldsymbol{\sigma}_{\text{<Customer\_ID='Customer\_ID' >}}(\text{Customer\_Purchase}))$

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

   Less_Than_5_Tuples <- SIGMA.book quantity < 5 (Store inventory)
   Less_Than_5_Joined_Tuples <- (Less_Than_5_Tuples) JOINS.DISBN = ISBN (Book)
   Less_Than_5_Titles/ISBN <- PI.Title, ISBN (Less_Than_5_Joined_Tuples))

   $\Pi_{\text{<ISBN, Title>}} (\boldsymbol{\sigma}_{\text{<Quantity < 5 >}}(\text{Store\_inventory}))$

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

   PratchetBooks <- SIGMA.Author = "Pratchet" (Book)
   PratchetBooks+CustomerID <- (PratchetBooks) JOINS.ISBN = ISBN (Buys)
   PratchetBooks+CustomerID+Customer_Name <- (PratchetBooks+CustomerID) JOINS.CustomerID = CustomerID (Customer_Name)
   NamesAndTitles <- PI.Name, Title (PratchetBooks+CustomerID+Customer_Name)

   $\Pi_{\text{<Customer Name, Title>}} ((\boldsymbol{\sigma}_{\text{<Author = 'Pratchett' >}} (\text{Books}) \bowtie_{\text{<ISBN = ISBN>}} (\text{Buys})) \bowtie_{\text{<CustomerId = CustomerId>}} (\text{Customer\_Name}))$

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

   $F_{\text{COUNT}}(\boldsymbol{\sigma}_{\text{<Customer\_ID = 'Customer\_ID'>}}(\text{Customer\_Purchase}))$

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

   $F_{\text{MAX}}({}_{\text{<Customer\_ID>}}F_{\text{COUNT}}(\text{Customer\_Purchase}) \bowtie_{\text{<Customer\_ID = 'Customer\_ID'>}}(\text{Customer}))$

4. Come up with three additional interesting queries that your database can provide. Give what the queries are supposed to retrieve in plain English and then as relational algebra. Your queries should include joins and at least one should include an aggregate function. At least one of your queries should use "extra" entities you added to your model in Checkpoint 01.

   - Count the number of books by a particular author
     - $F_{\text{COUNT}}(\boldsymbol{\sigma}_{\text{<Author = 'Brutus''>}}(\text{Book})) \bowtie_{\text{<ISBN >}}\text{Store\_Inventory})$
   - Find the total number of customers who have purchased a certain book
     - $F_{\text{COUNT}}(\boldsymbol{\sigma}_{\text{<ISBN = ISBN'>}}(\text{Book})) \bowtie_{\text{< Customer\_ID = 'Customer\_ID'>}} \text{Customer\_Purchase})$
   - Find the title of the most purchased book by George RR Martin
     - ${}_{\text{<Title>}}F_{\text{MAX}}({}_{\text{<ISBN, Title>}}F_{\text{COUNT(Receipt No.)}}(\text{Customer\_Purchase}) \bowtie_{\text{<Author= "George R.R. Martin">}}(\text{Book}))$

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.

- Supplier - (Supplier_ID, Location Address, Supplier Type, Name)
- Order - (Order_ID, OrderDate, orderTime, ISBN, Quantity, ShipmentStatus, Supplier_ID)
- Transaction - (Transaction_ID, Supplier_ID, date, time, amount, tax, payer, type)
- Book - (ISBN, title, year, price, publisher name, genre, Supplier_ID)
- Customer -(Customer_ID, Payment Info, Email)
- Customer_Purchase - (Receipt_No, ISBN, Title, Date_Time, Amount, Customer_ID, Quantity)
- Store_Inventory (DISBN, Book Quantity, ReOrder)
- Book_Author - (ISBN, AuthorId)
- AuthorList - (AuthorId, First, Last)
- Customer_Name - (Customer_ID, Customer Name)

CSE 3241 Project Checkpoint 03 – SQL and More SQL

Names	Date
Alfath Ahmed, Akshath Srinivasan, Satwinder Singh, Mark Maher	10-27-2020

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.**

- Supplier - (Supplier_ID, Location Address, Supplier Type, Name)
- Order - (Order_ID, OrderDate, orderTime, *ISBN*, Quantity, ShipmentStatus, *Supplier_ID*)
- Transaction - (Transaction_ID, *Supplier_ID*, date, time, amount, tax, payer, type)
- Book - (ISBN, title, year, price, publisher name, genre, *Supplier_ID*)
- Customer -(Customer_ID, Payment Info, Email)
- Customer_Purchase - (Receipt_No, *ISBN*, Title, Date_Time, Amount, *Customer_ID*, Quantity)
- Store_Inventory (DISBN, Book Quantity, ReOrder)
- Book_Author - (ISBN, AuthorId)
- AuthorList - (AuthorId, First, Last)
- Customer_Name - (Customer_ID, Customer Name)

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.

```
CREATE TABLE Supplier

(Supplier_ID          int             NOT NULL,

Location_Address      varchar(30)     NOT NULL,

Supplier_Type         varchar(30)     NOT NULL,

Name                  varchar(30)     NOT NULL,

PRIMARY KEY (Supplier_ID));
```

```
CREATE TABLE AuthorList

(AuthorId INTEGER NOT NULL,

First VARCHAR (15) NOT NULL,

 Last VARCHAR (15) NOT NULL,

PRIMARY KEY (AuthorId));


CREATE TABLE Transaction

(Transaction_ID          int              NOT NULL,

Supplier_ID              int              NOT NULL,

date                     timestamp        NOT NULL,

time                     time             NOT NULL,

amount                   int              NOT NULL,

tax                      int,

payer                    varchar(30)      NOT NULL,

type                     varchar(30)      NOT NULL,

PRIMARY KEY (Transaction_ID),

FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID);


CREATE TABLE Customer

(Customer_ID    int              NOT NULL,

Payment Info    varchar(30)      NOT NULL,

PRIMARY KEY(Customer_ID));
```

```
CREATE TABLE Store_Inventory

(DISBN                    int       NOT NULL,

Book Quantity             int,

ReOrder                   boolean        NOT NULL,

PRIMARY KEY(DISBN));


CREATE TABLE Order

(Order_ID       int            NOT NULL,

orderDate       timestamp      NOT NULL,

orderTime       time           NOT NULL,

ISBN            int            NOT NULL,

Quantity        int

Supplier_ID     int            NOT NULL,

FOREIGN KEY(Supplier_ID) REFERENCES Supplier(Supplier_ID));


CREATE TABLE Book

(ISBN                     int            NOT NULL,

title                     varchar(30)    NOT NULL,

year                      int            NOT NULL,

price                     int,

publisher name            varchar(30)    NOT NULL,

Supplier_ID               int            NOT NULL,

PRIMARY KEY(ISBN),

FOREIGN KEY(Supplier_ID) REFERENCES Supplier(Supplier_ID));
```

```
CREATE TABLE Customer_Purchase

(Receipt_No     int             NOT NULL,

ISBN            int             NOT NULL,

Title           varchar(30)     NOT NULL,

Date/Time       timestamp       NOT NULL,

Amount          int

Customer_ID     int             NOT NULL,

Quantity        int             NOT NULL,

PRIMARY KEY(Receipt_No),

FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID),

FOREIGN KEY(ISBN) REFERENCES Book(ISBN));


CREATE TABLE Shipment

(Shipment_ID            int                     NOT NULL,

Shipment Status         Boolean                 NOT NULL,

Content                         VARCHAR(30)             NOT NULL,

Supplier_ID             int                     NOT NULL,

PRIMARY KEY(Shipment_ID),

FOREIGN KEY(Supplier_ID) REFERENCES Supplier(Supplier_ID));


CREATE TABLE Buys

(Customer_ID    int     NOT NULL,

ISBN            int     NOT NULL,

FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID),
FOREIGN KEY(ISBN) REFERENCES Book(ISBN));


CREATE TABLE Book_Author

(ISBN           int             NOT NULL,

author          int             NOT NULL,

FOREIGN KEY(ISBN) REFERENCES Book(ISBN),

PRIMARY KEY(category));


CREATE TABLE Customer_Name

(Customer_ID            int             NOT NULL,

Customer Name           varchar(30)     NOT NULL,

FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID),

PRIMARY KEY(Customer_Name));
```

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 DISTINCT B.Title
      FROM Book B, Book_Author A
      WHERE A.Author = 'Pratchett' AND B.Price < 10 AND B.isbn = A.isbn;
      ```

   b. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)
      (Assuming: Customer_ID = 1)

      ```
      SELECT CP.Title, CP.Date_Time
      FROM Customer_Purchase CP
      WHERE Customer_ID = 1;
      ```

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

      ```
      SELECT B.Title, B.ISBN
      ```

```
FROM Store Inventory SI, Book B
WHERE SI.Book_Quantity < 5 AND SI.DISBN = B.ISBN;
```

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

```
SELECT CP.Customer_ID, CP.Title
FROM Customer_Purchase CP, Book B, Book_Author A
WHERE CP.ISBN = B.ISBN AND B.ISBN = A.ISBN AND A.Author = 'Pratchett';
```

e. Find the total number of books purchased by a single customer (you choose how to designate the customer)
(Assuming: Customer_ID = 1)

```
SELECT SUM(CP.Quantity)
FROM Customer_Purchase CP
WHERE CP.Customer_ID = 1;
```

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

```
FROM
        SELECT CP.Customer_ID, SUM(CP.Quantity) AS Total
        FROM Customer_Purchase CP
        WHERE
        GROUP BY CP.Customer_ID
        HAVING max(*);
```

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".

- Count the number of books by a particular author
  - $F_{\text{COUNT}}(\boldsymbol{\sigma}_{\text{<Author = 'Brutus''>}}(\text{Book})) \bowtie_{\text{<ISBN >}} \text{Store\_Inventory})$
- Find the total number of customers who have purchased a certain book
  - $F_{\text{COUNT}}(\boldsymbol{\sigma}_{\text{<ISBN = ISBN'>}}(\text{Book})) \bowtie_{\text{< Customer\_ID = 'Customer\_ID'>}} \text{Customer\_Purchase})$
- Find the title of the most purchased book by George RR Martin
  - ${}_{\text{<Title>}}F_{\text{MAX}}({}_{\text{<ISBN, Title>}}F_{\text{COUNT(Receipt No.)}}(\text{Customer\_Purchase}) \bowtie_{\text{<Author= "George R.R. Martin">}}(\text{Book}))$

```
SELECT SUM(A.QUANTITY)
```

```
FROM Book_Author A, Book B
WHERE A.ISBN = B.ISBN

(Assuming Title = 'Abnormally Large Krungus')
SELECT COUNT(*)
FROM Customer_Purchase CP
WHERE CP.Title = 'Abnormally Large Krungus';

SELECT MAX(COUNT(*))
FROM Customer_Purchase CP, Book_Author A
WHERE CP.ISBN = A.ISBN AND A.Author = 'George RR Martin'
GROUP BY CP.Title
HAVING
```

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,  Max(sum(CP.Amount))
   FROM Customer_purchase CP, customer C
   WHERE C.Customer_ID = CP.Customer_ID;
   ```

   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 C, Customer_Purchase CP
   WHERE C.Customer_ID = CP.Customer_ID
   GROUP BY C.Customer_ID
   HAVING CP.Amount > AVG(CP.Amount);
   ```

   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 CP.Title, SUM(CP.Quantity)
   FROM Customer_Purchase CP
   GROUP BY CP.Title
   ```

```
ORDER BY 2 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 CP.Title, SUM(CP.Amount)
FROM Customer_Purchase CP
GROUP BY CP.Title
ORDER BY 2 DESC;
```

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

```
SELECT A.Author
FROM Customer_Purchase CP, Book_Author A
WHERE CP.ISBN = A.ISBN
GROUP BY A.Author
HAVING MAX(SUM(CP.Quantity));
```

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.Author,
FROM Customer_Purchase CP, Book_Author A,
WHERE CP.ISBN = A.ISBN
GROUP BY A.Author
HAVING MAX(SUM(CP.Amount));
```

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 C
WHERE (
        SELECT A.Author,
        FROM Customer_Purchase CP, Book_Author A,
        WHERE CP.ISBN = A.ISBN
        GROUP BY A.Author
        HAVING MAX(SUM(CP.Amount));
```

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

```
SELECT A.Author,
FROM Book_Author A
WHERE
        SELECT C.name
        FROM Customer C, Customer_Purchase CP
        WHERE C.Customer_ID = CP.Customer_ID
        GROUP BY C.Customer_ID
        HAVING CP.Amount > AVG(CP.Amount);
```

CSE 3241 Project Checkpoint 04 – Functional Dependencies and Normal Forms

Names	Date
Alfath Ahmed, Akshath Srinivasan, Mark Maher, Satwinder Singh	11-15-2020

In a **NEATLY TYPED** document, provide the following:

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 03. **If you were instructed to change the model for Project Checkpoint 03, make sure you use the revised versions of your models.**
    - Supplier - (Supplier_ID, Location Address, Supplier Type, Name)
    - Order - (Order_ID, OrderDate, orderTime, Quantity, ShipmentStatus, *Supplier_ID*)
    - Transaction - (Transaction_ID, date, time, amount, tax, type, *Supplier_ID*)
    - Book - (ISBN, title, year, price, publisher name, genre, *Supplier_ID*)
    - Customer -(Customer_ID, Payment Info, Email)
    - Customer_Purchase - (Receipt_No, Date_Time, Cost, Quantity, *Customer_ID*)
    - Store_Inventory (DISBN, Book Quantity, ReOrder)
    - Book_Author - (ISBN, AuthorId)
    - AuthorList - (AuthorId, First, Last)
    - Customer_Name -(Customer_ID, Fname, Lname)

2. For each relation schema in your model, indicate the functional dependencies. Think carefully about what you are modeling here - make sure you consider all the possible dependencies in each relation and not just the ones from your primary keys. For example, a customer's credit card number is unique, and so will uniquely identify a customer even if you have another key in the same table (in fact, if the customer can have multiple credit card numbers, the dependencies can get even more involved).
    - **Supplier:**
      FD1 - {Supplier_ID} -> {Location_Address, Supplier_Type, Name}
    - **Order:**
      FD1 - {Order_ID} -> {OrderDate, OrderTime, Quantity, ShipmentStatus, Supplier_ID}
    - **Transaction:**
      FD1 - {Transaction_ID} -> {Date, Time, Amount, Tax, Quantity, Payer, Type}

- **Book:**
  FD1 - {ISBN} -> {title, year, publisher name, genre, supplier_ID}
- **Customer:**
  FD1 - {Customer_ID} -> {payment Info, Email}
- **Customer_Purchase:**
  FD1 - {Receipt_No, customer_ID} -> {Date_Time, cost, quantity}
- **Store_Inventory:**
  FD1 - {DISBN} -> {Book_Quantity, Reorder}
- **Book_Author:**
  FD1 - {ISBN} -> {AuthorId}
- **AuthorList:**
  FD1 - {AuthorId} -> {First, Last}
- **Customer_Name:**
  FD1 - {Customer_ID} -> {Fname, Lname}

3. For each relation schema in your model, determine the highest normal form of the relation. If the relation is not in 3NF, rewrite your relation schema so that it is in at least 3NF.
   - **CustomerPaymentInfo:**
     FD1 - {Customer_ID} -> {payment Info}
   - **Customer:**
     FD1 - {Customer_ID} -> {Email}

   Customer is split into CustomerPaymentInfo and CustomerEmail to normalize the relation into 3NF.

4. For each relation schema in your model that is in 3NF but not in BCNF, either rewrite the relation schema to BCNF or provide a short justification for why this relation should be an exception to the rule of putting relations into BCNF.
   Once Customer is split into two tables (CustomerPaymentInfo and CustomerEmail), the model is in 3NF as well as BCNF. Everything is in BCNF.

5. For your database, propose at least two interesting views that can be built from your relations. These views must involve joining at least two tables together each and must include some kind of aggregation in the view. Each view must also be able to be described by a one or two sentence description in plain English. Provide the code for constructing your views along with the English language description of what the view is supposed to be providing.

   Total Customer Bill

   - The following view explains the total customer bill.

```
CREATE VIEW TotalCustomerBill AS

        SELECT  N.id, N.fname, N.lname, sum(T.amount)

        FROM   Names N, Customer C, TransactionAmount
        WHERE N.id = C.id

        AND  T.epay = C.eid

GROUP BY N.id
```

Author Profit

- The following view explains the book author's profit amount.

```
CREATE VIEW AuthorProfitAmount AS

        SELECT N.fname, N.mname, N.lname, P.profit, x.valueSold

        FROM Names N,

                (SELECT O.bid, Sum (o.quantity) AS valueSold

                FROM OrderItems O

                GROUP BY O.BID) T,

                (SELECT T.Sum * B.price AS profit

                        FROM

                                (SELECT O.bid AS isbn, Sum(O.quantity) AS Sum

                                FROM OrderItems O

                                GROUP BY O.bid) T,

                                Book B

                        WHERE B.isbn = T.isbn

                ) P,

                BookAuthor BA

        WHERE BA.nameid = N.id
```