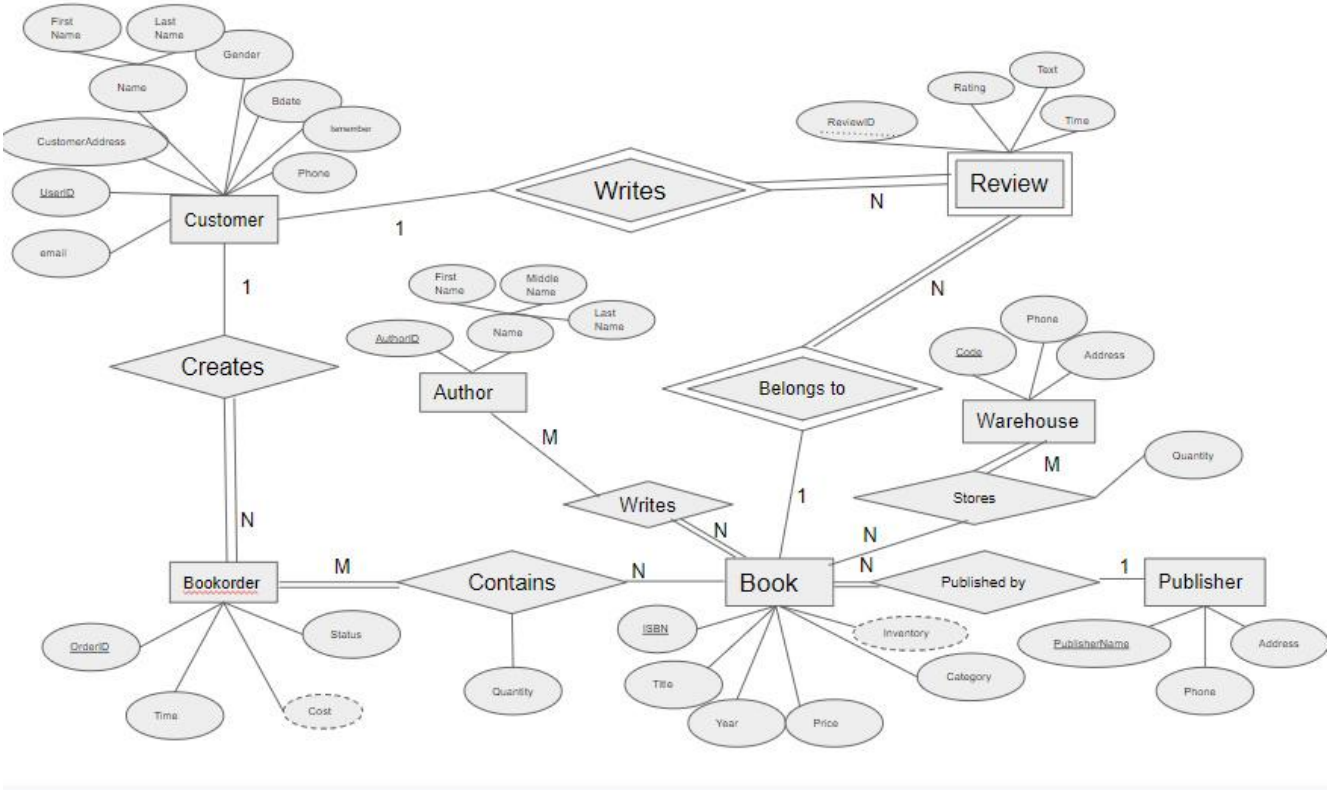
Section 1 - Database Description

1. ER-model



2. Relational Schema

Book (ISBN, Title, Year, Price, Category, PublisherName)

- Foreign key PublisherName References Publisher

Author (AuthorID, FirstName, MiddleName, LastName)

Customer (UserID, LastName, FirstName, Gender, Bdate, Phone, CustomerAddress, Email, IsMember)

Bookorder(OrderID, OrderTime, OrderStatus, UserID)

- Foreign key UserID References Customer

Publisher(PublisherName, Phone, Address)

Warehouse(Code, Address, Phone)

Review(ReviewID, UserID, ISBN, Rating, Text, Time)

- Foreign key UserID References Customer
- Foreign key ISBN References BOOK

CONTAINS(quantity, OrderID, ISBN)

- Foreign key OrderID References BookOrder

- Foreign key ISBN References BOOK

STORES(quantity, Code, ISBN)

- Foreign key Code references Warehouse
- Foreign key ISBN references Book.

WRITES (AuthorID, ISBN)

- Foreign key AuthorID references Author
- Foreign key ISBN references Book

3. Level of Normalization

BCNF:

Book (ISBN, Title, Year, Price, Category, PublisherName)
Author (AuthorID, FirstName, MiddleName, LastName)
Customer (UserID, LastName, FirstName, Gender, Bdate, Phone, CustomerAddress, Email, IsMember)
Bookorder(OrderID, Time, Status, UserID)
Review(ReviewID, UserID, ISBN, Rating, Text, Time)
CONTAINS(quantity, OrderID, ISBN)
STORES(quantity, Code, ISBN)
WRITES (AuthorID, ISBN)

2NF:

Publisher(PublisherName, Phone, Address)
Warehouse(Code, Address, Phone)

Why not BCNF?

Since publisher and warehouse are not in 3NF (Phone is dependent on Address, which is a nonprime attribute)

4. Description of Index

(1) Hash-based Indexes on Category
CREATE INDEX index1 ON Book (Category)
Rationale: Since ISBN are unique to each book, the number of ISBN is much larger than the number of Category. Thus, using Category as a hash-based index can make the execution more efficient and much faster.

(2) Hash-based Indexes on Order
CREATE INDEX index2 ON Order (Status);
Rationale: Since OrderID are unique to each order, the number of OrderID is much larger than the number of Status. Thus, using Status as a hash-based index can make the execution more efficient and much faster.

5. Description of Views

1. First view
    a. **English description:**
    List the titles and ISBNs for all books with less than 5 copies in stock

    b. Relational algebra:

    $ISBN\_SumQ \ <- \ \rho_{ISBN, Sum\_Quantity}(ISBN\ \mathcal{F}_{SUM\ Quantity}$ (Warehouse * Stores * Book)))

    $\pi_{title, ISBN}$ ($\sigma_{Sum\_Quantity > 5}$ (ISBN_SumQ * Book))

    c. SQL code:

    ```
    CREATE VIEW LessThan5Copies(Title, ISBN) AS
    SELECT Title, Book.ISBN
    FROM STORES, Book
    Where Book.ISBN=STORES.ISBN
    GROUP BY Book.ISBN
    HAVING SUM(quantity) < 5;
    ```

    d. Sample output:

	Title	ISBN
	Filter	Filter
1	How To Do Everything with Your Tablet PC	0072227710
2	SQL Server 2000 for Experienced DBA's	0072227885
3	The Data Warehouse Toolkit: The Complete Guide to ...	0471200247
4	Data Mining: Practical Machine Learning Tools and ...	1558605525

2. Second view
    a. **English description:**
    Find all books with rating greater than 4
    b. Relational algebra:

$\pi_{title, ISBN}(\sigma_{avg\_rating > 4}(\rho_{ISBN, avg\_rating}{}_{ISBN}\mathcal{F}_{AVERAGE\ Rating}(\text{Book} * \text{Review}))$

    c. SQL code:

```
CREATE VIEW GoodRatedBook (Title, ISBN) AS
SELECT Title, Book.ISBN

FROM Book, Review

WHERE Book.ISBN = Review.ISBN

GROUP BY Review.ISBN

HAVING AVG(Rating) > 4;
```

    d. Sample output:

	Title	ISBN
	Filter	Filter
1	Patron Saint of ...	0060540753
2	Twelve Times ...	0066214750
3	Investing in ...	0071414339
4	How To Do ...	0072227710
5	SQL Server 200...	0072227885
6	Analysis for ...	0072315318
7	Call of the Forest	0090044506
8	The Magician's ...	0156006219
9	The Guru's Guid...	0201615762
10	The Pianist	0312311354

6. Description of Transactions

1. **OSU has published a book called SQL from 0 to Give Up as a new publisher.**

```
BEGIN TRANSACTION ADD BOOK
        INSERT INTO Publisher
        VALUES ('OSU','6148886677', '281 W Lane Ave, Columbus, OH 43210');
```

```
        IF error THEN GO TO UNDO; END IF;
    INSERT INTO BOOK
    VALUES ('9999999999','SQL from 0 to Give Up', '2022', 32.41, 'Computer',OSU);
        IF error THEN GO TO UNDO; END IF;
    COMMIT;
    GO TO FINISH;
    UNDO:
      ROLLBACK;
    FINISH:
END TRANSACTION;
```

2. Add a new author whose name is Rock Rock Star to the book *SQL from 0 to Give Up*.

```
BEGIN TRANSACTION NEW_AUTHOR_NAME
    INSERT INTO AUTHOR
     ('5555555555','Rock','Rock','Star'),
    IF error THEN GO TO UNDO; END IF;
    INSERT INTO WRITES
    ('5555555555','9999999999'),

    IF error THEN GO TO UNDO; END IF;
    COMMIT;
    GO TO FINISH;
    UNDO:
      ROLLBACK;
    FINISH:
END TRANSACTION;
```

2. **User Darleen buys a book whose ISBN is 9999999999 in one order.**

```
BEGIN TRANSACTION DarleenMakesOrders
    INSERT INTO BOOKORDER
    VALUES ('1234567897', '1650124047','shopping', '9883084308' );
        IF error THEN GO TO UNDO; END IF;
     INSERT INTO CONTAINS
    VALUES (1, '1234567897', "9999999999" );
        IF error THEN GO TO UNDO; END IF;
    UPDATE BOOKORDER
    SET OrderStatus = 'pending';
    WHERE OrderStatus ='1234567897';
        IF error THEN GO TO UNDO; END IF;
```

```
        COMMIT;
        GO TO FINISH;
        UNDO:
            ROLLBACK;
        FINISH:
END TRANSACTION;
```

Section 2 - User Manual

1. Description of table and attribute

Book (ISBN, Title, Year, Price, Category, PublisherName)

- Table Book indicates the books in real world.
- ISBN: the unique code for each book
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- Title: the name of the book
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30 / NOT NULL
- Year: the year when the book was published
    - Data Type: CHAR
    - Constraints: length of 4 / NOT NULL
- Price: the price of the book
    - Data Type: REAL
    - Constraints: NOT NULL
- Category: the category each book belongs to
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 16 / NOT NULL
- PublisherName: the name of the publisher of the book
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 40 / NOT NULL

Author (AuthorID, FirstName, MiddleName, LastName)

- Table Author indicates the authors of books in real world.
- AuthorID: the unique code for each author
    - Data Type: CHAR
    - Constraints: length equal to 10 / NOT NULL
- FirstName: the first name of each author
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30 / NOT NULL
- MiddleName: The middle name of each author

    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30
- LastName: The last name of each author
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30 / NOT NULL

Customer (UserID, LastName, FirstName, Gender, Bdate, Phone, Address, Email, IsMember)

- Table Customer indicates the customer of the book store in the real world.
- UserID: the unique id of each customer
    - Data Type: CHAR
    - Constraints: length equal to 10 / NOT NULL
- LastName: the last name of each customer
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30 / NOT NULL
- FirstName: the first name of each author
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30 / NOT NULL
- Gender: the gender of each customer
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 10 / NOT NULL
- Bdate: the birthday of each customer
    - Data Type: DATE
- Phone: the phone numbere of each customer
    - Data Type: VARCHAR
    - Constraints: length equal to 10 / NOT NULL
- CustomerAddress: the address of each customer
    - Data Type: VARCHAR
    - Constraints: length equal to 30 / NOT NULL
- Email: the email address of each customer
    - Data Type: VARCHAR
    - Constraints: length equal to 30 / NOT NULL
- IsMember: the membership status of each customer
    - Data Type: BOOLEAN
    - Constraints: NOT NULL

Bookorder(OrderID, OrderTime, OrderStatus, UserID)

- Table Bookorder indicates the customer's order
- OrderID: the unique id of each order
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- OrderTime: the time that the order is made
    - Data Type: TIMESTAMP

    - Constraints: NOT NULL
- OrderStatus: the status of order
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 30 / NOT NULL
- UserID: the id of the user
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL

Review(<u>ReviewID,</u> <u>UserID</u>, <u>ISBN,</u> Rating, Text, Time)

- Review indicates the review of each book
- ReviewID: the unique id of each review
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- UserID: the unique id of the user
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- ISBN: the unique code for each book
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- Rating: the rate value of each book
    - Data Type: Decimal
    - Constraints: the value can have 2 digits overall and 1 digits to the right of the decimal point / NOT NULL
- Text: the review of each book
    - Data Type: VARCHAR
    - Constraints: length less than or equal to 150
- Time: the time when the review is written
    - Data Type: TIMESTMAP
    - Constraints: NOT NULL

CONTAINS(quantity, <u>OrderID</u>, <u>ISBN</u>)

- Table CONTAINS indicates which order contains which books
- Quantity : Orders can contain many of the same book each with an associated Quantity
    - Data Type: INT
    - Constraints: NOT NULL
- OrderID: the unique id of each order of customer
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- ISBN: the unique code for each book
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL

STORES (quantity, <u>Code</u>, <u>ISBN</u>)

- Table STORES indicates which warehouse stores which books
- Quantity : The warehouse can contain many of the same books each with an associated Quantity
    - Data Type: INT
    - Constraints: NOT NULL
- Code is a unique attribute that indicates the real-world book warehouse's identity code.
    - Data Type: CHAR
    - Constraints: length of 5 / NOT NULL
- ISBN: the unique code for each book
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL

WRITES (AuthorID, ISBN)

- Table WRITES indicates which author stores which books
- AuthorID: the unique code for each author
    - Data Type: CHAR
    - Constraints: length equal to 10 / NOT NULL
- ISBN: the unique code for each book
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL

Publisher(PublisherName, Phone, Address)

- Table publisher indicates the real-world book publishers.
- PublisherName: a unique attribute that indicates the real publisher's name.
    - Data Type: CHAR
    - Constraints: length of 40 / NOT NULL
- Phone is an attribute that indicates the publisher's phone number.
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- Address is an attribute that indicates the publisher's real address.
    - Data Type: CHAR
    - Constraints: length of 50 / NOT NULL

Warehouse(Code, Address, Phone)

- Table Warehouse indicates the real-world book warehouses.
- Code is a unique attribute that indicates the real-world book warehouse's identity code.
    - Data Type: CHAR
    - Constraints: length of 5 / NOT NULL
- Phone is an attribute that indicates the warehouse's phone number.
    - Data Type: CHAR
    - Constraints: length of 10 / NOT NULL
- Address is an attribute that indicates the warehouse's real address.
    - Data Type: CHAR

- Constraints: length of 50 / NOT NULL

2. Sample SQL queries

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

  a. $\pi_{title}(\sigma_{LastName=\text{“}Pratchett\text{”}\ AND\ Price<10}$(Author * Writes * Book))

  b. SELECT Title

  FROM Book, WRITES, Author

  WHERE Book.ISBN = WRITES.ISBN

  AND WRITES.AuthorID = Author.AuthorID

  AND LastName = 'Pratchett'

  AND Price < 10;

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

  a. $\pi_{title,\ time}(\sigma_{UserID=12345}$ (BookOrder * Contains *Book))

  b. SELECT Title, OrderTime

  FROM BookOrder, Customer, Book, Contains

  WHERE Customer.UserID = '19827469'

  AND BookOrder.UserID = Customer.UserID

  AND Contains.OrderID = BookOrder.OrderID

  AND Contains.ISBN = Book.ISBN;

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

  a. $ISBN\_SumQ \leftarrow \rho_{ISBN,\ Sum\_Quantity}(ISBN\ \mathcal{F}_{SUM\ Quantity}$ (Warehouse * Stores * Book)))

  $\pi_{title,\ ISBN}$ ($\sigma_{Sum\_Quantity>5}$ (ISBN_SumQ * Book))

b. SELECT Title, Book.ISBN

FROM STORES, Book

Where Book.ISBN=STORES.ISBN

GROUP BY Book.ISBN

HAVING SUM(quantity) < 5;

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

a. $\pi_{UserID, title}(\sigma_{LastName="Pratchett"}$ (BookOrder * Contains * Book * Writes * Author))

b. SELECT Customer.LastName, Customer.FirstName, Title

FROM Customer, BookOrder, Contains, Book, Writes, Author

WHERE Customer.UserID = BookOrder.UserID

AND BookOrder.OrderID = CONTAINS.OrderID

AND Contains.ISBN = Book.ISBN

AND Book.ISBN = Writes.ISBN

AND Writes.AuthorID = Author.AuthorID

AND Author.LastName = 'Pratchett';

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

a. $\mathcal{F}_{COUNT\ ISBN}(\sigma_{UserID=12345}$ (BookOrder * Contains *Book))

b. SELECT SUM(quantity)

FROM CUSTOMER, BOOKORDER, BOOK, CONTAINS

WHERE CUSTOMER.UserID = '19827469'

AND BOOKORDER.UserID = CUSTOMER.UserID

AND CONTAINS.OrderID = BOOKORDER.OrderID

AND CONTAINS.ISBN = BOOK.ISBN;

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

a. $TotalNum\_User \ <- \ \rho_{UserID,\ count} UserID\ \mathcal{F}_{count\ ISBN}$ (BookOrder * Contains *Book)))

$Max \ <- \ \rho_{count} \mathcal{F}_{MAX\ count\_books}\ TotalNum\_User$

Max * TotalNumbercUser

b. SELECT LastName, FirstName, SUM(Quantity)

FROM Customer, BOOKORDER, Contains

WHERE Customer.UserID = BOOKORDER.UserID

AND BOOKORDER.OrderID = Contains.OrderID

GROUP BY BOOKORDER.UserID

HAVING SUM(Quantity) = (

SELECT MAX(mycount)

FROM (

SELECT BOOKORDER.UserID, SUM(Quantity) AS mycount

FROM Customer, BOOKORDER, Contains

WHERE Customer.UserID = BOOKORDER.UserID

AND BOOKORDER.OrderID = Contains.OrderID

GROUP BY BOOKORDER.UserID)

);

- find titles of all books below 100 dollars which author is Simon Benninga

a. $\pi_{title}(\sigma_{FirstName="Simon" \text{ AND } LastName="Benninga" \text{ AND } Price < 100}$(Author * Writes * Book))

b. SELECT Title
FROM Author A, Writes W, Book B
WHERE A.AuthorID = W.AuthorID
AND W.ISBN = B.ISBN
AND A.FirstName = 'Simon'
AND A.LastName = 'Benninga'
AND Price < 100;

- find titles of all books published in 83 Lukken Alley

a. $\pi_{title}(\sigma_{Address="83\ Lukken\ Alley"}$ (Book*Publisher))

b. SELECT title
FROM (Book JOIN Publisher ON Book.PublisherName =Publisher.PublisherName)
WHERE Address ='83 Lukken Alley';

- find all books with rating greater than 4

a. $\pi_{title}(\sigma_{avg\_rating > 4}(\rho_{ISBN,\ avg\_rating} ISBN\ \mathcal{F}_{AVERAGE\ Rating}$(Book * Review))

b. SELECT Title

FROM Book, Review

WHERE Book.ISBN = Review.ISBN

GROUP BY Review.ISBN

HAVING AVG(Rating) > 4;

3. 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.

INSERT INTO Publisher (PublisherName, Phone, Address)
VALUES ('OSU','6143740000','1900 CANNON DR');

INSERT INTO Author (AuthorID, FirstName, MiddleName, LastName)
VALUES ('1231231233','Parker', NULL, 'Wiksell');

INSERT INTO Book (ISBN, title, year, price, category, publishername)
VALUES ('1234567891', 'SQL from 0 to giveup', '2021','32.41', 'textbook', 'OSU');

INSERT INTO WRITES (AuthorID,ISBN)
VALUES ('1231231233','1234567891');

INSERT INTO Customer (UserID, LastName, FirstName, Gender, Bdate, Phone, CustomerAddress, Email, IsMember)
VALUES ('9883084999','Dar','Tilt','Male','1968-07-30','999-860-4945','10 Porter Pass','dtilt0@sciencedaily.com','true');

The foreign key Book.PublisherName references Publisher, so we need to insert Publisher before Book.
The foreign key Writes.AuthorID references Author, so we need to insert Author before Writes.
The foreign key Writes.ISBN references Book, so we need to insert Book before Writes.

4. 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.

(1) Delete books

To delete a book from the book table, we need to delete all the tuples taking ISBN as a foreign key(stores, contains, review and writes) first.

Then we can delete a specific book by its ISBN from book table

DELETE FROM STORES

```
WHERE ISBN = '1234567891';

DELETE FROM CONTAINS
WHERE ISBN = '1234567891';

DELETE FROM Review
WHERE ISBN = '1234567891';

DELETE FROM Writes
WHERE AuthorID = '1231231233'
AND ISBN = '1234567891';

DELETE FROM Book
WHERE ISBN = '1234567891';
```

(2) Delete author

To delete an author from the author table, we need to delete all the tuples taking AuthorID as foreign key(writes).

Since we deleted it above, we can directly delete the author by the AuthorID from the author table.

```
DELETE FROM Author
WHERE AuthorID = '1231231233';
```

(3) Delete Customer

To delete a customer from the customer table, we need to delete all the tuples taking UserID as a foreign key(Review, Bookorder).

```
DELETE FROM Review
WHERE UserID = '9883084308';

DELETE FROM Bookorder
WHERE UserID = '9883084308';
```

```
DELETE FROM Customer
WHERE UserID = '9883084308';
```

(4) Delete Publisher

To delete a publisher from the publisher table, we need to delete all the tuples taking PublisherName as a foreign key.

Since there is no table taking PublsiherName as a foreign key, we can directly delete the publisher by the PublisherName from the publisher table.

```
DELETE FROM Publisher
WHERE PublisherName = 'OSU';
```

Appendix: All the Checkpoints:

The green highlight indicates our revision

1. Original checkpoint1:

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.

Entities	Attributes
Book	• ISBN • Title • Year • Price • Category • Inventory • Author
Customer	• UserID • LastName • FirstName • Gender • Date of birth • Phone number • Address • Email • IsPrime
Publisher	• Name • Phone • Address

2. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, “CUSTOMER entities purchase BOOK entities”).

CUSTOMER entities purchase BOOK entities.
BOOK entities are purchased by CUSTOMER entities.

BOOK entities are published by PUBLISHER entities.
PUBLISHER entities publish BOOK entities.

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.

Entities	Attributes
Review（weak)	• ID • Rating • Text • Time
Order	• OrderID • Date • Time • Cost • Status

CUSTOMER entities create ORDER entities / ORDER entities contain BOOK entities.
Adding the ORDER entities can allow us to keep track of more information about the purchase.

REVIEW entities belong to BOOK entities / REVIEW entities is written by CUSTOMER entities
Adding the REVIEW entities can offer detailed information about the books to help users make decisions.

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.

- 1. Retrieve the order status of a specific customer
- 2. List all the books in the inventory
- 3. List all the publishers in alphabetical order
- 4. List all the books with decreasing price sort with one specific author
- 5. List the quantity of the books in the shopping cart for one specific customer.
- 6. Retrieve the average rating for one specific book

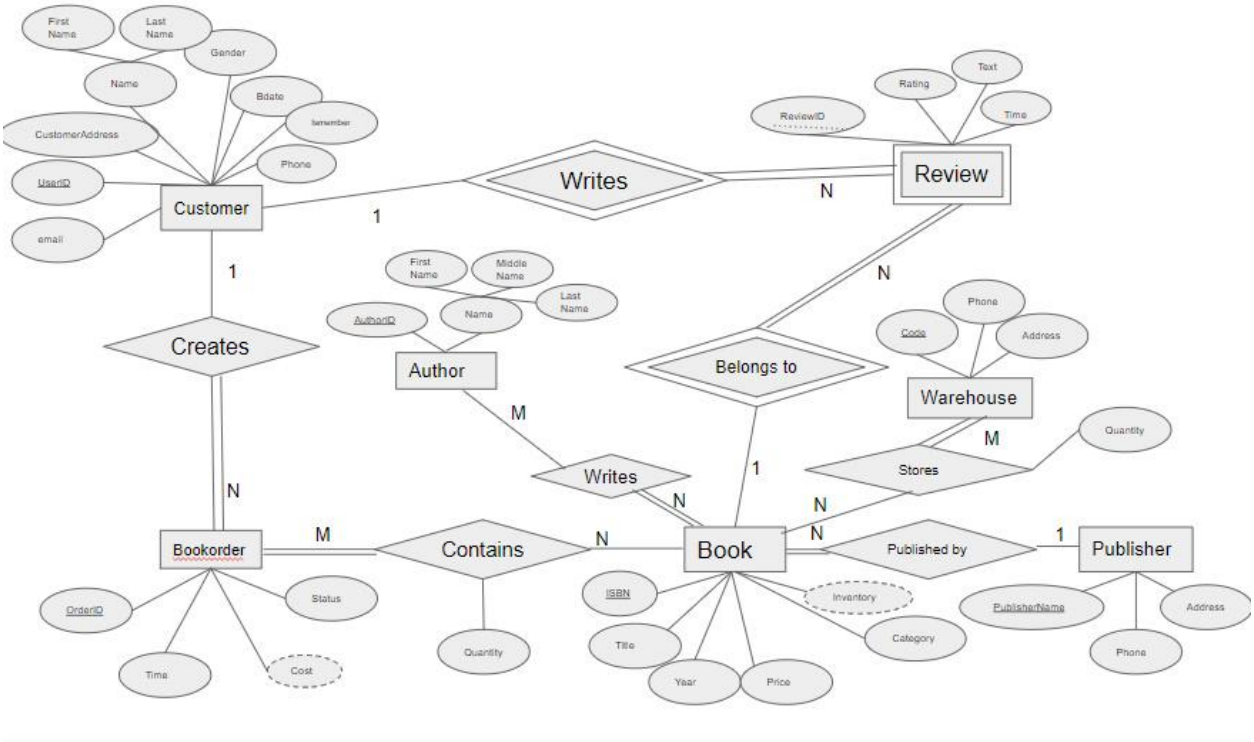
5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you've outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your model to allow for publishers to be added as separate entities.
Publisher entities publish book entities.

Since publisher name is a primary key, we can add a new publisher to the database by simply adding its unique publisher name.
Yes, since the publisher is one independent entity.

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.

1. Modify the book's price
2. Change the customer's membership
3. Modify the publisher's phone number
4. Change the order's status
5. Add the book's review rating.

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.



2. Revised checkpoint1:

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.

Entities	Attributes
Book	• ISBN • Title • Year • Price • Category •
Author	• AuthorID • FirstName • MiddleName • LastName
Customer	• UserID • LastName • FirstName • Gender • Bdate • Phone • Address • Email • IsMember
Publisher	• PublisherName • Phone • Address
BookOrder	• OrderID • Date • Time • Cost • Status

2. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, “CUSTOMER entities purchase BOOK entities”).

BOOK entities are published by PUBLISHER entities.
PUBLISHER entities publish BOOK entities.

CUSTOMER entities create BOOKORDER entities
ORDER entities are created by CUSTOMER entities

BOOKORDER entities contains BOOK entities
BOOK entities are contained by ORDER entities

AUTHOR entities write BOOK entities.
BOOK entities are written by AUTHOR entities.

BOOK entities are written by AUTHOR entities
AUTHOR entities write BOOK entities

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.

Entities	Attributes
Review（weak)	• ID • Rating • Text • Time
Warehouse	• Code • Address • Phone

CUSTOMER entities create BOOKORDER entities / BOOKORDER entities contain BOOK entities.
Adding the BOOKORDER entities can allow us to keep track of more information about the purchase.

REVIEW entities belong to BOOK entities / REVIEW entities is written by CUSTOMER entities
Adding the REVIEW entities can offer detailed information about the books to help users make decisions.

WAREHOUSE entities store BOOK entities / BOOK entities are stored in the WAREHOUSE entities.
Adding the WAREHOUSE entities allow us to track the stock of the BOOK entities.

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.

- 1. Retrieve the order status of a specific customer
- 2. List all the books in the inventory
- 3. List all the publishers in alphabetical order
- 4. List all the books with decreasing price sort with one specific author
- 5. List the quantity of the books in the shopping cart for one specific customer.
- 6. Retrieve the average rating for one specific book
- 7. List the phone number of a warehouse.

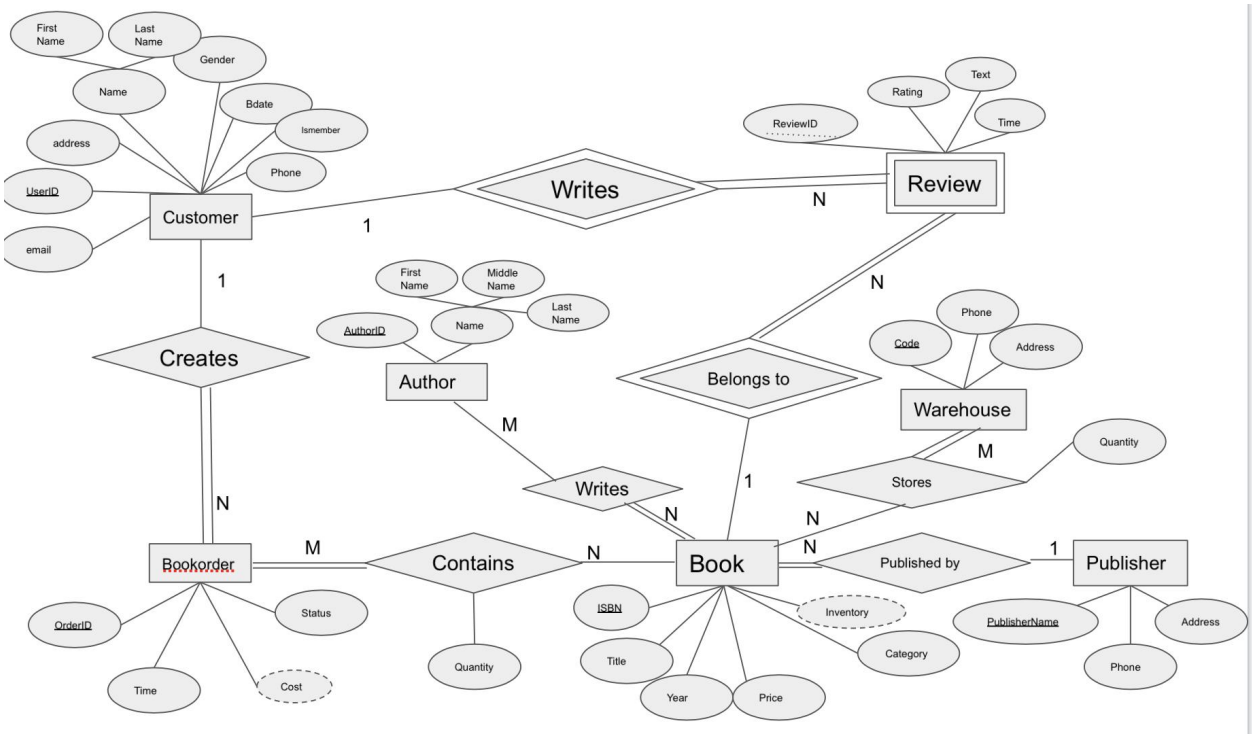
5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you've outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your model to allow for publishers to be added as separate entities. Publisher entities publish book entities.

Since publisher name is a primary key, we can add a new publisher to the database by simply adding its unique publisher name.
Yes, since the publisher is one independent entity.

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.

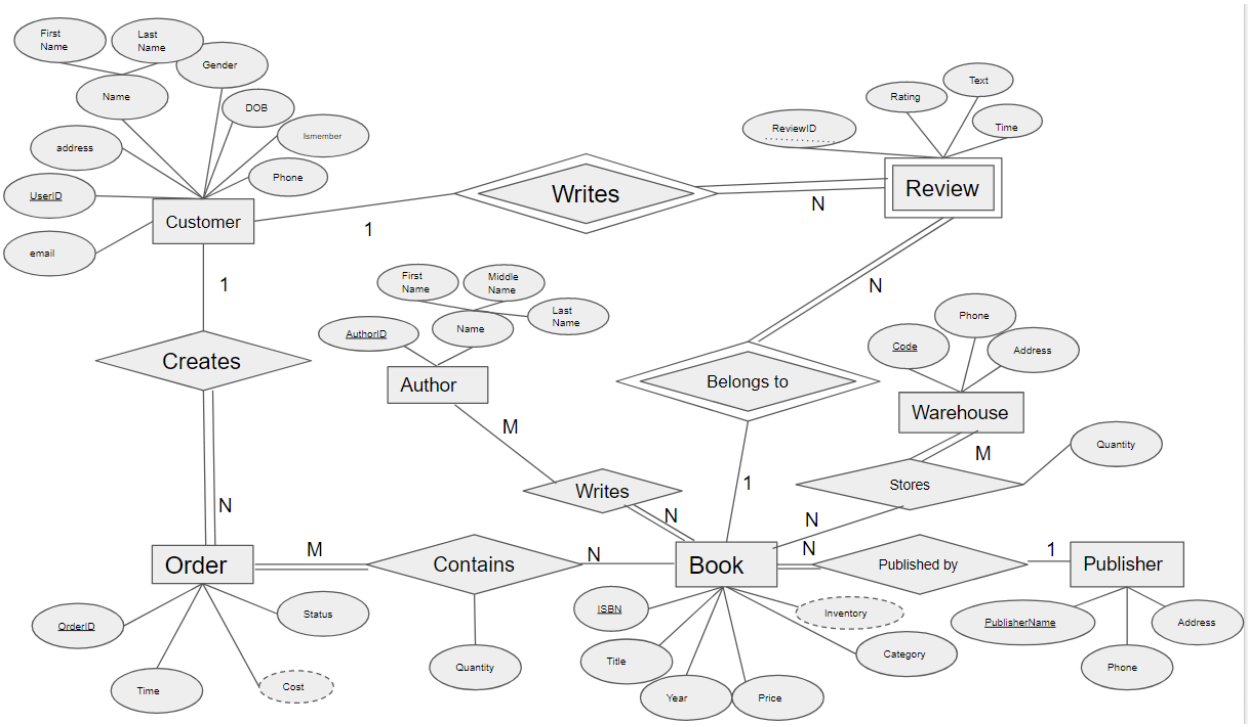
1. Modify the book's price
2. Change the customer's membership
3. Modify the publisher's phone number
4. Change the order's status
5. Add the book's review rating.
6. Change the phone number of the warehouse

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.



3. Original checkpoint2:

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.



2. Map your ER model to a relational schema. Indicate all primary and foreign keys.

   Book (ISBN, Title, Year, Price, Category, PublisherName)
   - Foreign key PublisherName References Publisher

   Author (AuthorID, FirstName, MiddleName, LastName)

   Customer (UserID, LastName, FirstName, Gender, Date of birth, Phone, Address, Email, IsMember)

   Order (OrderID, Time, Status, UserID)
   - Foreign key UserID References Customer

   Publisher(PublisherName, Phone, Address)

   Warehouse(Code, Address, Phone)

   Review(ID, Rating, Text, Time)

   CONTAINS(quantity, OrderID, ISBN)
   - Foreign key OrderID References Order
   - Foreign key ISBN References BOOK

   STORES(quantity, Code, ISBN)
   - Foreign key Code references Warehouse

- Foreign key ISBN references Book.

WRITES (AuthorID, ISBN)

- Foreign key AuthorID references Author
- Foreign key ISBN references Book

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

   $\pi_{title}(\sigma_{LastName=\text{“}Pratchett\text{”}\ AND\ Price<10}$(Author * Writes * Book))

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

   $\pi_{title,\ time}(\sigma_{UserID=12345}$ (Order * Contains *Book))

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

   $ISBN\_SumQ\ <-\ \rho_{ISBN,\ Sum\_Quantity}(ISBN\ \mathcal{F}_{SUM\ Quantity}$ (Warehouse * Stores * Book)))

   $\pi_{title,\ ISBN}$ ($\sigma_{Sum\_Quantity>5}$ (ISBN_SumQ * Book))

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

   $\pi_{UserID,\ title}$ ($\sigma_{LastName=\text{“}Pratchett\text{”}}$ (Order * Contains * Book * Writes * Author))

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

   $\mathcal{F}_{COUNT\ ISBN}$ ($\sigma_{UserID=12345}$ (Order * Contains *Book))

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

   $TotalNum\_User\ <-\ \rho_{UserID,\ count}UserID\ \mathcal{F}_{count\ ISBN}$ (Order * Contains *Book)))

   $Max\ <-\ \rho_{count}\mathcal{F}_{MAX\ count\_books}\ TotalNum\_User$

Max * TotalNumberUser

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.

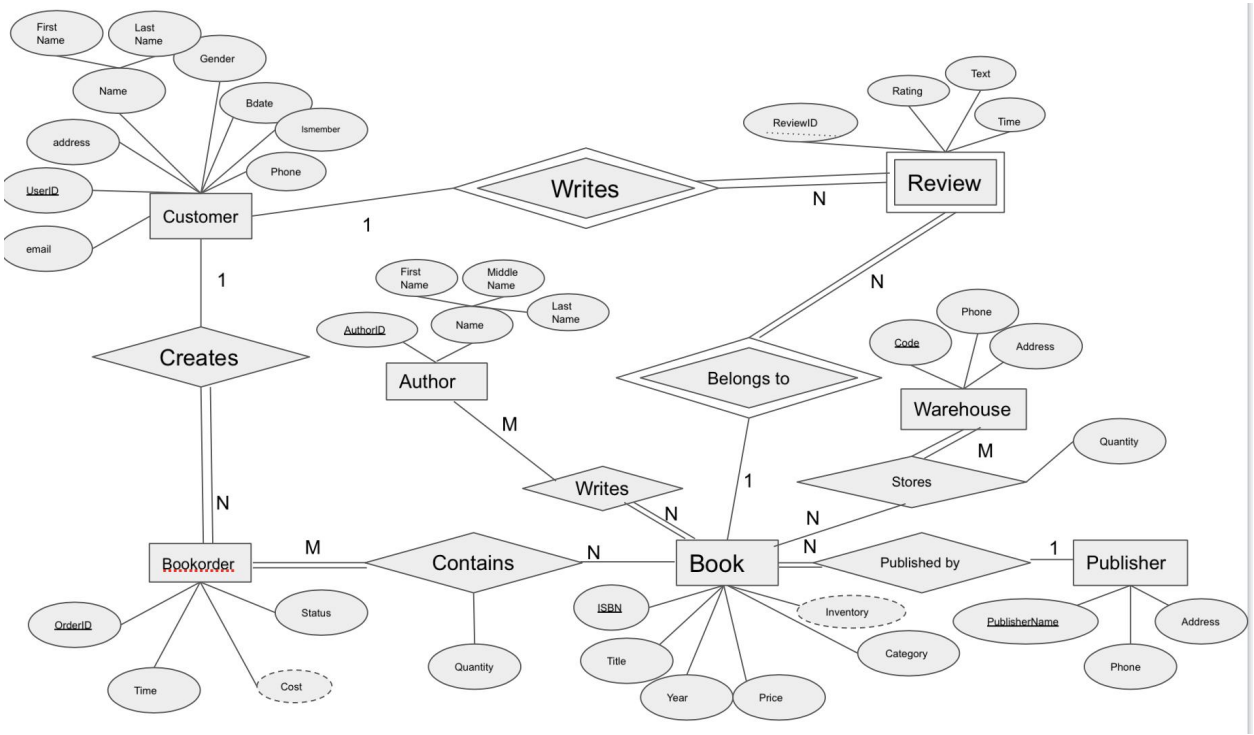
    - find titles of all books below 100 dollars which author is Parker Wiksell
        - $\pi_{title}(\sigma_{FirstName="Parker" \text{ AND } LastName="Wiksell" \text{ AND } Price < 100}$(Author * Writes * Book))
    - find all books published by The OSU Press
        - $\sigma_{PublisherName="The\ OSU\ Press"}$ (Book)
    - find all books with rating greater than 4
        - $\pi_{title}(\sigma_{avg\_rating > 4}(\rho_{ISBN,\ avg\_rating} ISBN\ \mathcal{F}_{AVERAGE\ Rating}$(Book * Review))

4. Revised checkpoint2:

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.



2. Map your ER model to a relational schema. Indicate all primary and foreign keys.

Book (ISBN, Title, Year, Price, Category, PublisherName)

- Foreign key PublisherName References Publisher

Author (AuthorID, FirstName, MiddleName, LastName)

Customer (UserID, LastName, FirstName, Gender, Bdate, Phone, Address, Email, IsMember)

BookOrder (OrderID, Time, Status, UserID)

- Foreign key UserID References Customer

Publisher(PublisherName, Phone, Address)

Warehouse(Code, Address, Phone)

Review(UserID, ReviewID, ISBN, Rating, Text, Time)

CONTAINS(quantity, OrderID, ISBN)

- Foreign key OrderID References Order
- Foreign key ISBN References BOOK

STORES(quantity, Code, ISBN)

- Foreign key Code references Warehouse
- Foreign key ISBN references Book.

WRITES (AuthorID, ISBN)

- Foreign key AuthorID references Author
- Foreign key ISBN references Book

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

   $\pi_{title}(\sigma_{LastName=\text{“}Pratchett\text{”}\ AND\ Price<10}$ (Author * Writes * Book))

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

   $\pi_{title,\ time}(\sigma_{UserID=12345}$ (BookOrder * Contains *Book))

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

   $ISBN\_SumQ \ <-\ \rho_{ISBN,\ Sum\_\ Quantity}(ISBN\ \mathcal{F}_{SUM\ Quantity}$ (Warehouse * Stores * Book)))

   $\pi_{title,\ ISBN}$ ($\sigma_{Sum\_\ Quantity>5}$ (ISBN_SumQ * Book))

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

   $\pi_{UserID,\ title}$ ($\sigma_{LastName=\text{“}Pratchett\text{”}}$ (BookOrder * Contains * Book * Writes * Author))

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

   $\mathcal{F}_{COUNT\ ISBN}$ ($\sigma_{UserID=12345}$ (BookOrder * Contains *Book))

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

   $TotalNum\_User \ <-\ \rho_{UserID,\ count}\ UserID\ \mathcal{F}_{count\ ISBN}$ (BookOrder * Contains *Book)))

   $Max \ <-\ \rho_{count}\mathcal{F}_{MAX\ count\_\ books}\ TotalNum\_User$

   Max * TotalNumberUser

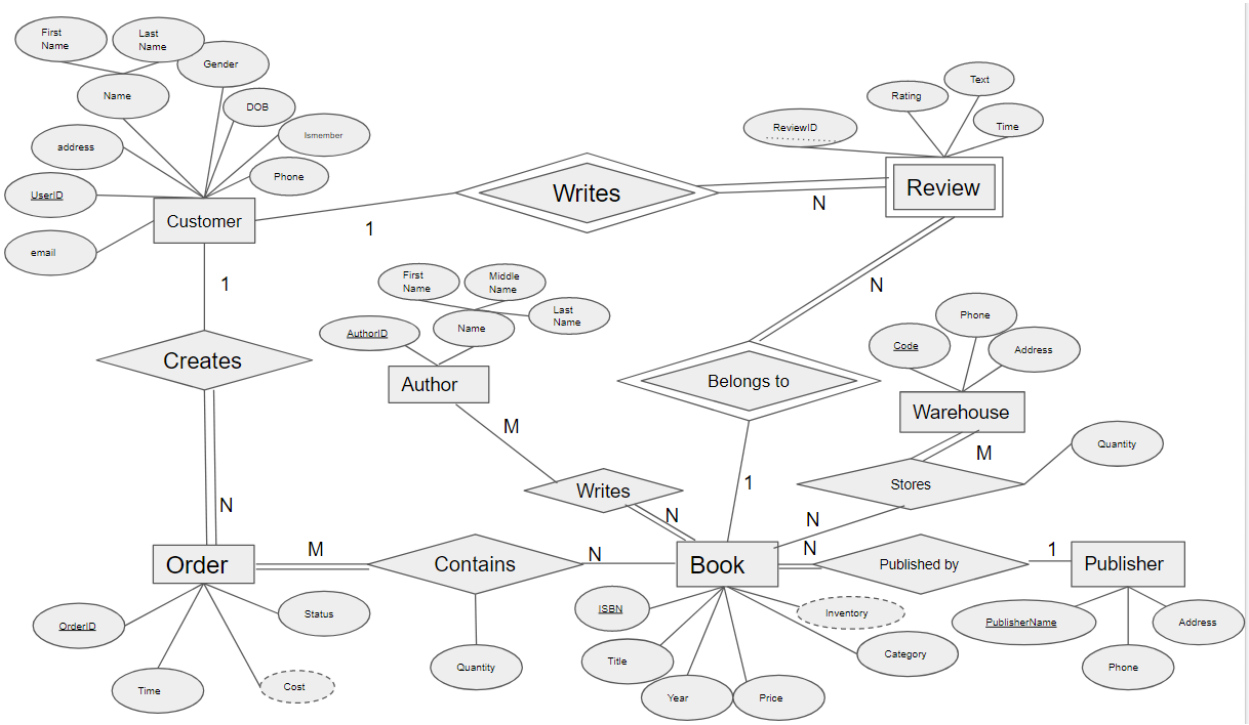
4. Come up with three additional interesting queries that your database can provide. Give what the queries are supposed to retrieve in plain English and then as relational algebra. Your queries should include joins and at least one should include an aggregate function. At least one of your queries should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

5. find titles of all books below 100 dollars which author is Simon Benninga
   a. $\pi_{title}(\sigma_{FirstName="Simon"\ AND\ LastName="Benninga"\ AND\ Price<100}$(Author * Writes * Book))
6. find titles of all books published in 83 Lukken Alley
   a. $\pi_{title}(\sigma_{Address="83\ Lukken\ Alley"}$(Book*Publisher))
7. find all books with rating greater than 4
   a. $\pi_{title}(\sigma_{avg\_rating>4}(\rho_{ISBN,\ avg\_rating}{}_{ISBN}\mathcal{F}_{AVERAGE\ Rating}$(Book * Review))

8. Original checkpoint3:

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, Title, Year, Price, Category, PublisherName)

- Foreign key PublisherName References Publisher

Author (AuthorID, FirstName, MiddleName, LastName)

Customer (UserID, LastName, FirstName, Gender, Date of birth, Phone, Address, Email, IsMember)

Bookorder(OrderID, Time, Status, UserID)

- Foreign key UserID References Customer

Publisher(PublisherName, Phone, Address)

Warehouse(Code, Address, Phone)

Review(ReviewID, UserID, ISBN, Rating, Text, Time)

- Foreign key UserID References Customer
- Foreign key ISBN References BOOK

CONTAINS(quantity, OrderID, ISBN)

- Foreign key OrderID References Bookorder
- Foreign key ISBN References BOOK

STORES(quantity, Code, ISBN)

- Foreign key Code references Warehouse
- Foreign key ISBN references Book.

WRITES (AuthorID, ISBN)

- Foreign key AuthorID references Author
- Foreign key ISBN references Book

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”:

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

```
SELECT Title
```

```
FROM Book, WRITES,  Author

WHERE Book.ISBN = WRITES.ISBN

AND WRITES.AuthorID = Author.AuthorID

AND LastName = '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, OrderTime

FROM BookOrder, Customer, Book, Contains

WHERE Customer.UserID = '19827469'

AND BookOrder.UserID = Customer.UserID

AND Contains.OrderID = BookOrder.OrderID

AND Contains.ISBN = Book.ISBN;
```

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

```
SELECT Title, Book.ISBN

FROM STORES, Book

Where Book.ISBN=STORES.ISBN

GROUP BY Book.ISBN

HAVING SUM(quantity) < 5;
```

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

```
SELECT Customer.LastName, Customer.FirstName, Title
```

```
FROM Customer, BookOrder, Contains, Book, Writes, Author

WHERE Customer.UserID = BookOrder.UserID

AND BookOrder.OrderID = CONTAINS.OrderID

AND Contains.ISBN = Book.ISBN

AND Book.ISBN = Writes.ISBN

AND Writes.AuthorID = Author.AuthorID

AND Author.LastName = 'Pratchett';
```

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

```
SELECT SUM(quantity)

FROM CUSTOMER, BOOKORDER, BOOK, CONTAINS

WHERE CUSTOMER.UserID = '19827469'

AND BOOKORDER.UserID = CUSTOMER.UserID

AND CONTAINS.OrderID = BOOKORDER.OrderID

AND CONTAINS.ISBN = BOOK.ISBN;
```

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

```
SELECT LastName, FirstName, SUM(Quantity)

        FROM Customer, BOOKORDER, Contains

        WHERE Customer.UserID = BOOKORDER.UserID

AND BOOKORDER.OrderID = Contains.OrderID

GROUP BY BOOKORDER.UserID

HAVING SUM(Quantity) = (
```

```
SELECT MAX(mycount)
FROM (
        SELECT BOOKORDER.UserID, SUM(Quantity) AS mycount
        FROM Customer, BOOKORDER, Contains
        WHERE Customer.UserID = BOOKORDER.UserID
        AND BOOKORDER.OrderID = Contains.OrderID
        GROUP BY BOOKORDER.UserID)
);
```

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".

（1）find titles of all books below 100 dollars which author is Simon Benninga

$\pi_{title}(\sigma_{FirstName="Simon"\ AND\ LastName="Benninga"\ AND\ Price<100}$(Author * Writes * Book))

```
SELECT Title
FROM Author A, Writes W, Book B
WHERE A.AuthorID = W.AuthorID
AND W.ISBN = B.ISBN
AND A.FirstName = 'Simon'
AND A.LastName = 'Benninga'
AND Price < 100;
```

（2）find titles of all books published in 83 Lukken Alley

$\pi_{title}(\sigma_{Address="83\ Lukken\ Alley"}$ (Book*Publisher))

```
SELECT title
FROM (Book JOIN Publisher ON Book.PublisherName =Publisher.PublisherName)
WHERE Address ='83 Lukken Alley';
```

(3) find all books with rating greater than 4

$\pi_{title}(\sigma_{avg\_rating > 4}(\rho_{ISBN,\, avg\_rating} ISBN\, \mathcal{F}_{AVERAGE\ Rating}(\text{Book} * \text{Review}))$

```
SELECT Title

FROM Book, Review

WHERE Book.ISBN = Review.ISBN

GROUP BY Review.ISBN

HAVING AVG(Rating) > 4;
```

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 LastName, FirstName, SUM(price*quantity) as COST

FROM BOOKORDER, Contains, Book,Customer

WHERE Customer.UserID = BOOKORDER.UserID

AND BOOKORDER.OrderID = Contains.OrderID

AND Contains.ISBN = Book.ISBN

GROUP BY BOOKORDER.UserID;
```

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

```
SELECT  LastName, FirstName, Email,COST

FROM (SELECT LastName, FirstName, Email,SUM(price*quantity) as COST

                        FROM BOOKORDER, Contains, Book,Customer

                        WHERE Customer.UserID = BOOKORDER.UserID

                        AND BOOKORDER.OrderID = Contains.OrderID

                        AND Contains.ISBN = Book.ISBN

                        GROUP BY BOOKORDER.UserID),

   (SELECT AVG(COST) AS avg_cost

   FROM (SELECT Customer.UserID,SUM(price*quantity) as COST

                        FROM BOOKORDER, Contains, Book,Customer

                        WHERE Customer.UserID = BOOKORDER.UserID

                        AND BOOKORDER.OrderID = Contains.OrderID

                        AND Contains.ISBN = Book.ISBN

                        GROUP BY BOOKORDER.UserID))

    WHERE COST>avg_cost;
```

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 Title, SUM(C.Quantity) as total_qty_sold

FROM Contains C, Book B

Where C.ISBN = B.ISBN

GROUP BY 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 Title, SUM(revenue_individual) as total_revenue

FROM

        (SELECT Title, Quantity*Price as revenue_individual

        FROM Contains C, Book B

        WHERE C.ISBN = B.ISBN)

GROUP BY Title

ORDER BY 2;
```

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

```
                SELECT FirstName,LastName

                FROM

                        (SELECT Author.FirstName,Author.LastName,
        SUM(Quantity) as total_qty_sold

                        FROM Contains, Book, Writes, Author

                        Where Contains.ISBN = Book.ISBN

                        AND Book.ISBN = Writes.ISBN

                        AND Writes.AuthorID = Author.AuthorID

                        GROUP by Author.AuthorID) AS Author_sold,

                (SELECT MAX(total_qty_sold) as max

                FROM

                        (SELECT Writes.AuthorID, SUM(Quantity) as
        total_qty_sold
```

```
        FROM Contains, Book, Writes, Author
        Where Contains.ISBN = Book.ISBN
        AND Book.ISBN = Writes.ISBN
        AND Writes.AuthorID = Author.AuthorID
        GROUP by Author.AuthorID) AS Author_sold) as max_qty
    Where Author_sold.total_qty_sold=max_qty.max;
```

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

```
SELECT AUTHOR_revenue.FirstName,AUTHOR_revenue.LastName
FROM
(SELECT A.FirstName,A.LastName, SUM(Quantity*Price) as
revenue_individual
FROM Contains C, Book B, Writes W, Author A
WHERE C.ISBN = B.ISBN
AND B.ISBN = W.ISBN
AND W.AuthorID = A.AuthorID
GROUP BY A.AuthorID) as AUTHOR_revenue,
(SELECT Max(revenue_individual) as most
    FROM
    (SELECT A.AuthorID, SUM(Quantity*Price) as revenue_individual
    FROM Contains C, Book B, Writes W, Author A
    WHERE C.ISBN = B.ISBN
    AND B.ISBN = W.ISBN
```

```
        AND W.AuthorID = A.AuthorID

        GROUP BY A.AuthorID))most_revenue

    WHERE most_revenue.most=AUTHOR_revenue.revenue_individual
```

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

```
SELECT DISTINCT(Customer.UserID), Customer.LastName,
Customer.FirstName
FROM Customer, BOOKORDER, Contains, Book, Writes, Author
WHERE BOOKORDER.UserID = Customer.UserID
AND Contains.OrderID = BOOKORDER.OrderID
AND Contains.ISBN = Book.ISBN
AND Writes.ISBN = Book.ISBN
AND Writes.AuthorID = Author.AuthorID
AND Author.AuthorID =
    (SELECT AUTHOR_revenue.AuthorID FROM
        (SELECT A.FirstName,A.LastName, A.AuthorID,
    SUM(Quantity*Price) as revenue_individual

        FROM Contains C, Book B, Writes W, Author A

        WHERE C.ISBN = B.ISBN

        AND B.ISBN = W.ISBN

        AND W.AuthorID = A.AuthorID

        GROUP BY A.AuthorID) as AUTHOR_revenue,

        (SELECT Max(revenue_individual) as most

        FROM

            (SELECT A.AuthorID, SUM(Quantity*Price) as
    revenue_individual

            FROM Contains C, Book B, Writes W, Author A
```

```
                                    WHERE C.ISBN = B.ISBN

                                    AND B.ISBN = W.ISBN

                                    AND W.AuthorID = A.AuthorID

                                    GROUP BY A.AuthorID))most_revenue

                              WHERE
                        most_revenue.most=AUTHOR_revenue.revenue_individual

                );
```

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

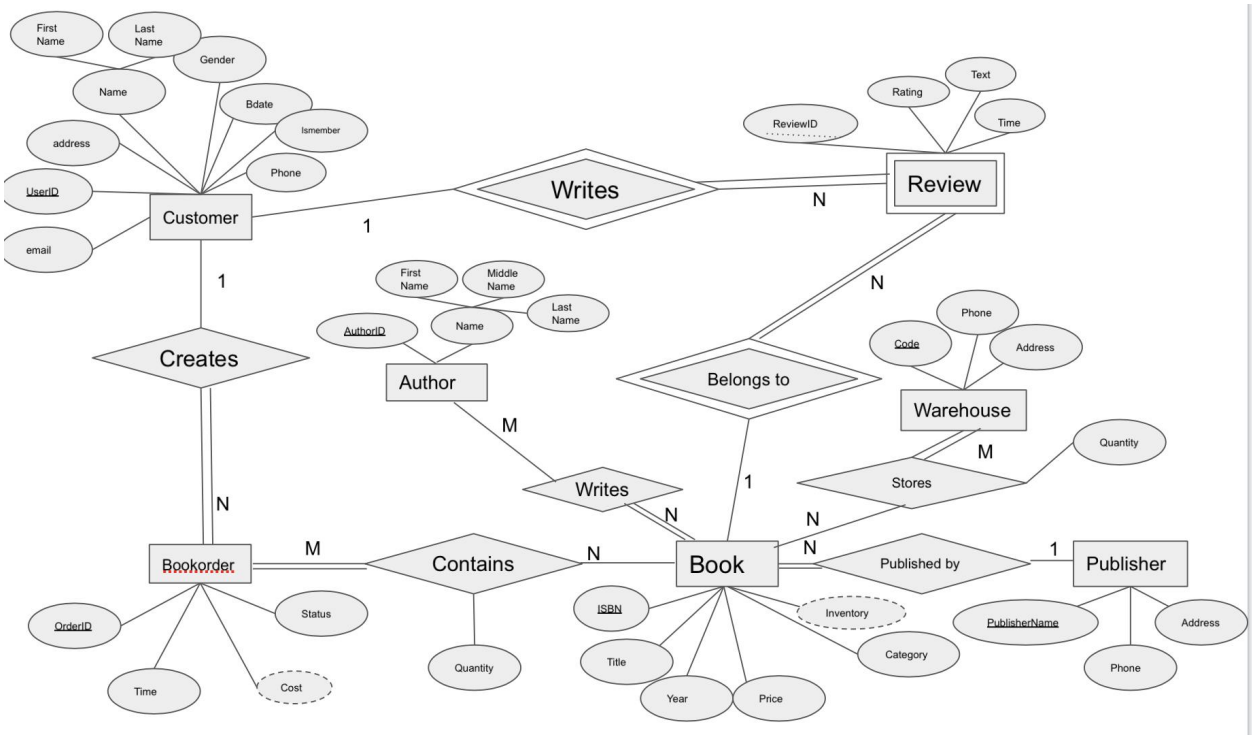
```
SELECT DISTINCT *
FROM
(SELECT Customer.UserID, Author.FirstName, Author.LastName
FROM Author, writes, book, CONTAINS,BOOKORDER,Customer,
(SELECT  UserID
FROM (SELECT Customer.UserID,SUM(price*quantity) as COST
                        FROM BOOKORDER, Contains, Book,Customer
                        WHERE Customer.UserID = BOOKORDER.UserID
                        AND BOOKORDER.OrderID = Contains.OrderID
                        AND Contains.ISBN = Book.ISBN
                        GROUP BY BOOKORDER.UserID),
   (SELECT AVG(COST) AS avg_cost
   FROM (SELECT Customer.UserID,SUM(price*quantity) as COST
                        FROM BOOKORDER, Contains, Book,Customer
                        WHERE Customer.UserID = BOOKORDER.UserID
                        AND BOOKORDER.OrderID = Contains.OrderID
                        AND Contains.ISBN = Book.ISBN
                        GROUP BY BOOKORDER.UserID))
    WHERE COST>avg_cost) as more_than_average

WHERE Author.AuthorID = WRITES.AuthorID
AND WRITES.ISBN = Book.ISBN
AND BOOKORDER.UserID=Customer.UserID
AND CONTAINS.OrderID=BOOKORDER.OrderID
AND CONTAINS.ISBN=Book.ISBN
AND more_than_average.UserID=Customer.UserID);
```

9. Revised checkpoint3:

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, Title, Year, Price, Category, PublisherName)

- Foreign key PublisherName References Publisher

Author (AuthorID, FirstName, MiddleName, LastName)

Customer (UserID, LastName, FirstName, Gender, Bdate, Phone, Address, Email, IsMember)

Bookorder(OrderID, Time, Status, UserID)

- Foreign key UserID References Customer

Publisher(PublisherName, Phone, Address)

Warehouse(Code, Address, Phone)

Review(ReviewID, UserID, ISBN, Rating, Text, Time)

- Foreign key UserID References Customer
- Foreign key ISBN References BOOK

CONTAINS(quantity, OrderID, ISBN)

- Foreign key OrderID References Bookorder
- Foreign key ISBN References BOOK

STORES(quantity, Code, ISBN)

- Foreign key Code references Warehouse
- Foreign key ISBN references Book.

WRITES (AuthorID, ISBN)

- Foreign key AuthorID references Author
- Foreign key ISBN references Book

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":

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

```
SELECT Title

FROM Book, WRITES,  Author

WHERE Book.ISBN = WRITES.ISBN

AND WRITES.AuthorID = Author.AuthorID

AND LastName = '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, OrderTime

FROM BookOrder, Customer, Book, Contains
```

```
WHERE Customer.UserID = '19827469'

AND BookOrder.UserID = Customer.UserID

AND Contains.OrderID = BookOrder.OrderID

AND Contains.ISBN = Book.ISBN;
```

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

```
SELECT Title, Book.ISBN

FROM STORES, Book

Where Book.ISBN=STORES.ISBN

GROUP BY Book.ISBN

HAVING SUM(quantity) < 5;
```

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

```
SELECT Customer.LastName, Customer.FirstName, Title

FROM Customer, BookOrder, Contains, Book, Writes, Author

WHERE Customer.UserID = BookOrder.UserID

AND BookOrder.OrderID = CONTAINS.OrderID

AND Contains.ISBN = Book.ISBN

AND Book.ISBN = Writes.ISBN

AND Writes.AuthorID = Author.AuthorID

AND Author.LastName = 'Pratchett';
```

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

```
SELECT SUM(quantity)
FROM CUSTOMER, BOOKORDER, BOOK, CONTAINS
WHERE CUSTOMER.UserID = '19827469'
AND BOOKORDER.UserID = CUSTOMER.UserID
AND CONTAINS.OrderID = BOOKORDER.OrderID
AND CONTAINS.ISBN = BOOK.ISBN;
```

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

```
SELECT LastName, FirstName, SUM(Quantity)
        FROM Customer, BOOKORDER, Contains
        WHERE Customer.UserID = BOOKORDER.UserID
AND BOOKORDER.OrderID = Contains.OrderID
GROUP BY BOOKORDER.UserID
HAVING SUM(Quantity) = (
SELECT MAX(mycount)
FROM (
        SELECT BOOKORDER.UserID, SUM(Quantity) AS mycount
        FROM Customer, BOOKORDER, Contains
        WHERE Customer.UserID = BOOKORDER.UserID
        AND BOOKORDER.OrderID = Contains.OrderID
        GROUP BY BOOKORDER.UserID)
);
```

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".

（1）find titles of all books below 100 dollars which author is Simon Benninga

$\pi_{title}(\sigma_{FirstName="Simon" \text{ AND } LastName="Benninga" \text{ AND } Price < 100}$（Author * Writes * Book))

```
SELECT Title
FROM Author A, Writes W, Book B
WHERE A.AuthorID = W.AuthorID
AND W.ISBN = B.ISBN
AND A.FirstName = 'Simon'
AND A.LastName = 'Benninga'
AND Price < 100;
```

（2）find titles of all books published in 83 Lukken Alley

$\pi_{title}(\sigma_{Address="83\ Lukken\ Alley"}$ (Book*Publisher))

```
SELECT title
FROM (Book JOIN Publisher ON Book.PublisherName =Publisher.PublisherName)
WHERE Address ='83 Lukken Alley';
```

（3）find all books with rating greater than 4

$\pi_{title}(\sigma_{avg\_rating > 4}(\rho_{ISBN,\ avg\_rating}{}_{ISBN}\mathcal{F}_{AVERAGE\ Rating}$(Book * Review))

```
SELECT Title

FROM Book, Review

WHERE Book.ISBN = Review.ISBN

GROUP BY Review.ISBN

HAVING AVG(Rating) > 4;
```

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 LastName, FirstName, SUM(price*quantity) as COST

FROM BOOKORDER, Contains, Book,Customer

WHERE Customer.UserID = BOOKORDER.UserID

AND BOOKORDER.OrderID = Contains.OrderID

AND Contains.ISBN = Book.ISBN

GROUP BY BOOKORDER.UserID;
```

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

```
SELECT  LastName, FirstName, Email,COST

FROM (SELECT LastName, FirstName, Email,SUM(price*quantity) as COST

                    FROM BOOKORDER, Contains, Book,Customer

                    WHERE Customer.UserID = BOOKORDER.UserID

                    AND BOOKORDER.OrderID = Contains.OrderID

                    AND Contains.ISBN = Book.ISBN

                    GROUP BY BOOKORDER.UserID),

  (SELECT AVG(COST) AS avg_cost
```

```
FROM (SELECT Customer.UserID,SUM(price*quantity) as COST
                    FROM BOOKORDER, Contains, Book,Customer
                    WHERE Customer.UserID = BOOKORDER.UserID
                    AND BOOKORDER.OrderID = Contains.OrderID
                    AND Contains.ISBN = Book.ISBN
                    GROUP BY BOOKORDER.UserID))
 WHERE COST>avg_cost;
```

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 Title, SUM(C.Quantity) as total_qty_sold
FROM Contains C, Book B
Where C.ISBN = B.ISBN
GROUP BY 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 Title, SUM(revenue_individual) as total_revenue
FROM
        (SELECT Title, Quantity*Price as revenue_individual
        FROM Contains C, Book B
        WHERE C.ISBN = B.ISBN)
GROUP BY 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 FirstName,LastName
FROM
    (SELECT Author.FirstName,Author.LastName,
SUM(Quantity) as total_qty_sold
    FROM Contains, Book, Writes, Author
    Where Contains.ISBN = Book.ISBN
    AND Book.ISBN = Writes.ISBN
    AND Writes.AuthorID = Author.AuthorID
    GROUP by Author.AuthorID) AS Author_sold,
(SELECT MAX(total_qty_sold) as max
FROM
    (SELECT Writes.AuthorID, SUM(Quantity) as
total_qty_sold
    FROM Contains, Book, Writes, Author
    Where Contains.ISBN = Book.ISBN
    AND Book.ISBN = Writes.ISBN
    AND Writes.AuthorID = Author.AuthorID
    GROUP by Author.AuthorID) AS Author_sold) as max_qty
  Where Author_sold.total_qty_sold=max_qty.max;
```

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

```
SELECT AUTHOR_revenue.FirstName,AUTHOR_revenue.LastName
FROM

(SELECT A.FirstName,A.LastName, SUM(Quantity*Price) as
revenue_individual

FROM Contains C, Book B, Writes W, Author A

WHERE C.ISBN = B.ISBN

AND B.ISBN = W.ISBN

AND W.AuthorID = A.AuthorID

GROUP BY A.AuthorID) as AUTHOR_revenue,

(SELECT Max(revenue_individual) as most

    FROM

    (SELECT A.AuthorID, SUM(Quantity*Price) as revenue_individual

    FROM Contains C, Book B, Writes W, Author A

    WHERE C.ISBN = B.ISBN

    AND B.ISBN = W.ISBN

    AND W.AuthorID = A.AuthorID

    GROUP BY A.AuthorID))most_revenue

WHERE most_revenue.most=AUTHOR_revenue.revenue_individual
```

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

```
SELECT DISTINCT(Customer.UserID), Customer.LastName,
Customer.FirstName
```

```
FROM Customer, BOOKORDER, Contains, Book, Writes, Author
WHERE BOOKORDER.UserID = Customer.UserID
AND Contains.OrderID = BOOKORDER.OrderID
AND Contains.ISBN = Book.ISBN
AND Writes.ISBN = Book.ISBN
AND Writes.AuthorID = Author.AuthorID
AND Author.AuthorID =
        (SELECT AUTHOR_revenue.AuthorID FROM
                (SELECT A.FirstName,A.LastName, A.AuthorID,
        SUM(Quantity*Price) as revenue_individual

                FROM Contains C, Book B, Writes W, Author A

                WHERE C.ISBN = B.ISBN

                AND B.ISBN = W.ISBN

                AND W.AuthorID = A.AuthorID

                GROUP BY A.AuthorID) as AUTHOR_revenue,

                (SELECT Max(revenue_individual) as most

                FROM

                        (SELECT A.AuthorID, SUM(Quantity*Price) as
        revenue_individual

                        FROM Contains C, Book B, Writes W, Author A

                        WHERE C.ISBN = B.ISBN

                        AND B.ISBN = W.ISBN

                        AND W.AuthorID = A.AuthorID

                        GROUP BY A.AuthorID))most_revenue

                WHERE
        most_revenue.most=AUTHOR_revenue.revenue_individual

);
```

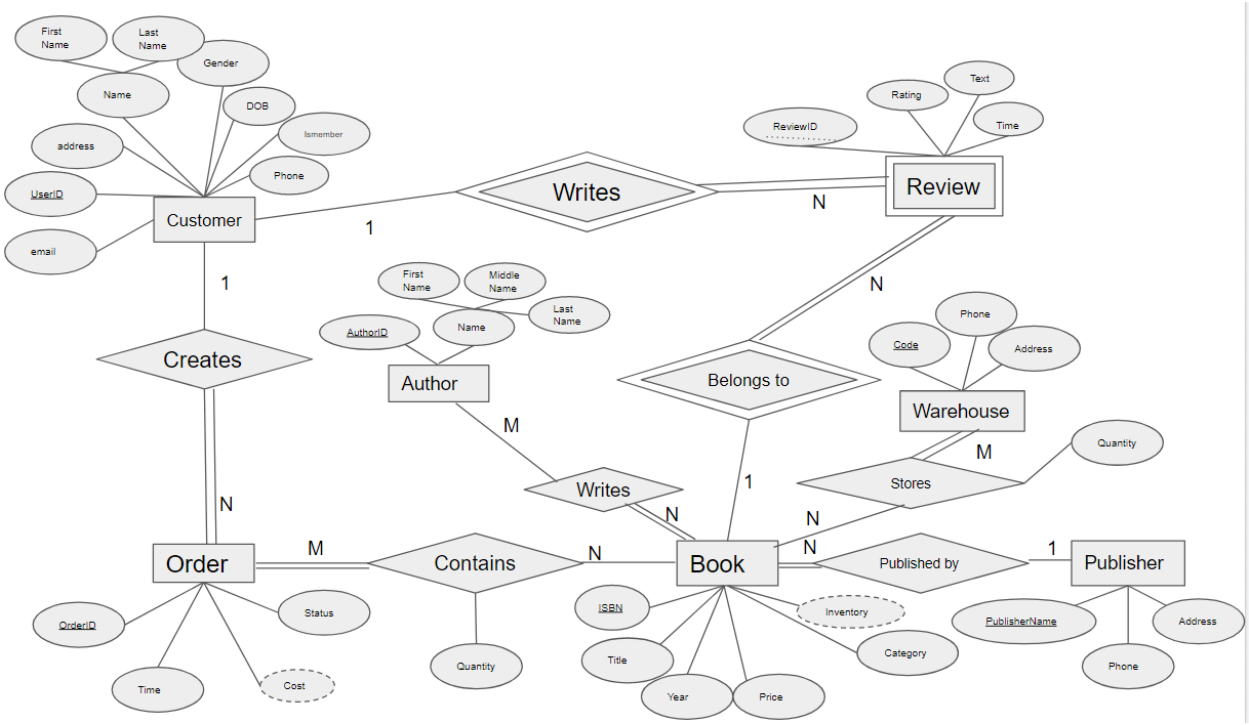
h. Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```
SELECT DISTINCT *
FROM
(SELECT Customer.UserID, Author.FirstName, Author.LastName
FROM Author, writes, book, CONTAINS,BOOKORDER,Customer,
(SELECT  UserID
FROM (SELECT Customer.UserID,SUM(price*quantity) as COST
                        FROM BOOKORDER, Contains, Book,Customer
                        WHERE Customer.UserID = BOOKORDER.UserID
                        AND BOOKORDER.OrderID = Contains.OrderID
                        AND Contains.ISBN = Book.ISBN
                        GROUP BY BOOKORDER.UserID),
   (SELECT AVG(COST) AS avg_cost
   FROM (SELECT Customer.UserID,SUM(price*quantity) as COST
                        FROM BOOKORDER, Contains, Book,Customer
                        WHERE Customer.UserID = BOOKORDER.UserID
                        AND BOOKORDER.OrderID = Contains.OrderID
                        AND Contains.ISBN = Book.ISBN
                        GROUP BY BOOKORDER.UserID))
    WHERE COST>avg_cost) as more_than_average

WHERE Author.AuthorID = WRITES.AuthorID
AND WRITES.ISBN = Book.ISBN
AND BOOKORDER.UserID=Customer.UserID
AND CONTAINS.OrderID=BOOKORDER.OrderID
AND CONTAINS.ISBN=Book.ISBN
AND more_than_average.UserID=Customer.UserID);
```

10. Original checkpoint4:

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.



Book (ISBN, Title, Year, Price, Category, PublisherName)

- Foreign key PublisherName References Publisher

Author (AuthorID, FirstName, MiddleName, LastName)

Customer (UserID, LastName, FirstName, Gender, Date of birth, Phone, Address, Email, IsMember)

Bookorder(OrderID, Time, Status, UserID)

- Foreign key UserID References Customer

Publisher(PublisherName, Phone, Address)

Warehouse(Code, Address, Phone)

Review(ReviewID, UserID, ISBN, Rating, Text, Time)

- Foreign key UserID References Customer
- Foreign key ISBN References BOOK

CONTAINS(quantity, OrderID, ISBN)

- Foreign key OrderID References Bookorder
- Foreign key ISBN References BOOK

STORES(quantity, Code, ISBN)

- Foreign key Code references Warehouse
- Foreign key ISBN references Book.

WRITES (AuthorID, ISBN)

- Foreign key AuthorID references Author
- Foreign key ISBN references Book

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).

Book: ISBN -> {Title, Year, Price, Category, PublisherName}

Author: AuthorID -> {FirstName, MiddleName, LastName}

Customer: UserID -> {LastName, FirstName, Gender, Date of birth, Phone, Address, Email, IsMember}

Bookorder: OrderID -> {Time, Status, UserID}

Publisher: PublisherName ->{Phone, Address}, Address ->Phone

Warehouse: Code -> {Address, Phone}, Address ->Phone

Review: {ReviewID, UserID} -> {ISBN, Rating, Text, Time}

CONTAINS: {OrderID, ISBN}-> quantity

STORES: {Code, ISBN}-> quantity

WRITES: {AuthorID, ISBN}->{AuthorID, ISBN}

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.

All of them are in 3NF

3NF:

Book (ISBN, Title, Year, Price, Category, PublisherName)
Author (AuthorID, FirstName, MiddleName, LastName)
Customer (UserID, LastName, FirstName, Gender, Date of birth, Phone, Address, Email, IsMember)
Bookorder(OrderID, Time, Status, UserID)
Review(ReviewID, UserID, ISBN, Rating, Text, Time)
CONTAINS(quantity, OrderID, ISBN)
STORES(quantity, Code, ISBN)

WRITES (AuthorID, ISBN)

NOT 3NF:

Publisher(PublisherName, Phone, Address)
Warehouse(Code, Address, Phone)

-> Change to 3NF:

PublisherAddress(PublisherName, Address)
PublisherPhone(Address, Phone)

Warehouse(Code, Address)
WarehousePhone (Address, Phone)

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.

All are 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.

1. First view
    a. List all customer names and e-mail addresses for customers who have spent less than the average customer.
    b. SQL code:

```
CREATE VIEW LessSpentCustomer (Lname, Fname, Email)
SELECT  LastName, FirstName, Email
FROM (SELECT LastName, FirstName, Email,SUM(price*quantity) as COST
                        FROM BOOKORDER, Contains, Book,Customer
                        WHERE Customer.UserID = BOOKORDER.UserID
                        AND BOOKORDER.OrderID = Contains.OrderID
                        AND Contains.ISBN = Book.ISBN
                        GROUP BY BOOKORDER.UserID),
  (SELECT AVG(COST) AS avg_cost
  FROM (SELECT Customer.UserID,SUM(price*quantity) as COST
                        FROM BOOKORDER, Contains, Book,Customer
                        WHERE Customer.UserID = BOOKORDER.UserID
                        AND BOOKORDER.OrderID = Contains.OrderID
                        AND Contains.ISBN = Book.ISBN
```

```
                    GROUP BY BOOKORDER.UserID))
    WHERE COST<avg_cost;
```

2. Second view
    a. Find the least popular author in the database
    b. SQL code:

```
CREATE VIEW LeastPopularAuthor (Fname, Lname)
SELECT FirstName,LastName
FROM
        (SELECT Author.FirstName,Author.LastName, SUM(Quantity) as
total_qty_sold
        FROM Contains, Book, Writes, Author
        Where Contains.ISBN = Book.ISBN
        AND Book.ISBN = Writes.ISBN
        AND Writes.AuthorID = Author.AuthorID
        GROUP by Author.AuthorID) AS Author_sold,
(SELECT MIN(total_qty_sold) as min
FROM
        (SELECT Writes.AuthorID, SUM(Quantity) as total_qty_sold
        FROM Contains, Book, Writes, Author
        Where Contains.ISBN = Book.ISBN
        AND Book.ISBN = Writes.ISBN
        AND Writes.AuthorID = Author.AuthorID
        GROUP by Author.AuthorID) AS Author_sold) as min_qty
    Where Author_sold.total_qty_sold=min_qty.min;
```