Part 1

These five key strategies, each linked to certain class features in your design, can enhance data correctness and integrity throughout your classes and help enforce data quality in the implementation of class diagrams for the Real Books Online Bookshop.

- **Data Validation in Attributes**

Validation checks should be used to ensure correct formatting for properties like email in the Customer class and ISBN in the Book class.

Email and ISBN formats may be checked using regular expressions to make sure they are properly formatted before data is stored. To verify that the email is formatted correctly, for example, the Customer class may include a validateEmail() function in the addToCart() or placeOrder() methods.

- **Attribute Constraints**

To avoid negative values, constraints should be used for characteristics such as stockQuantity in the Book class or quantity in the CartItem and OrderItem classes.

To impose a minimum value of zero, addItem(Book, quantity: int) and updateStock(quantity: int) functions should be protected. These limitations guarantee that amounts stay reasonable and accurately represent ordered quantities or stock levels.

- **Consistent Data Types and Formats**

To maintain uniformity throughout the system, the orderDate attribute in the Order class and the paymentDate attribute in the Payment class should always use the same date format.

All date-related data is kept consistent by implementing a single date format (YYYY-MM-DD, for example). Accurate computations and comparisons are made possible by using standardised data types for every attribute across classes (for example, float for pricing and int for quantities). This helps to avoid mismatched formats.

- **Dependency Constraints Between Classes**

To guarantee that the status of an order is accurately represented, dependencies between Cart, Order, and OrderItem are enforced.

For example, only when the Cart contains items should the Order class permit creation. To make sure the Cart isn't empty, a dependency check may be added to placeOrder(). The Order class may also verify if the payment was successful before changing the order state to "completed."

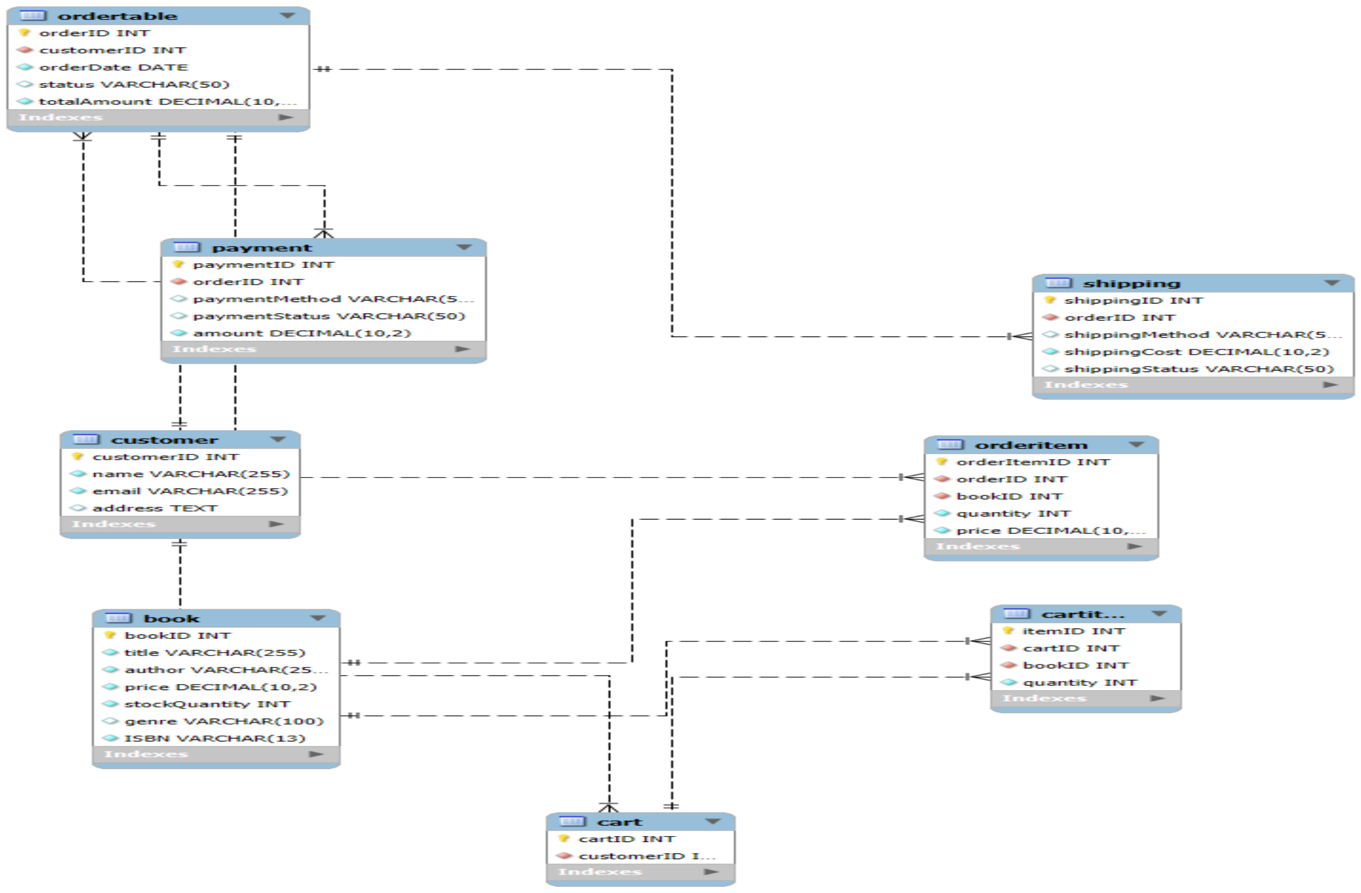
Range and Value Constraints

To avoid irrational pricing, the price attribute in the Book and OrderItem classes has to contain a positive value requirement.

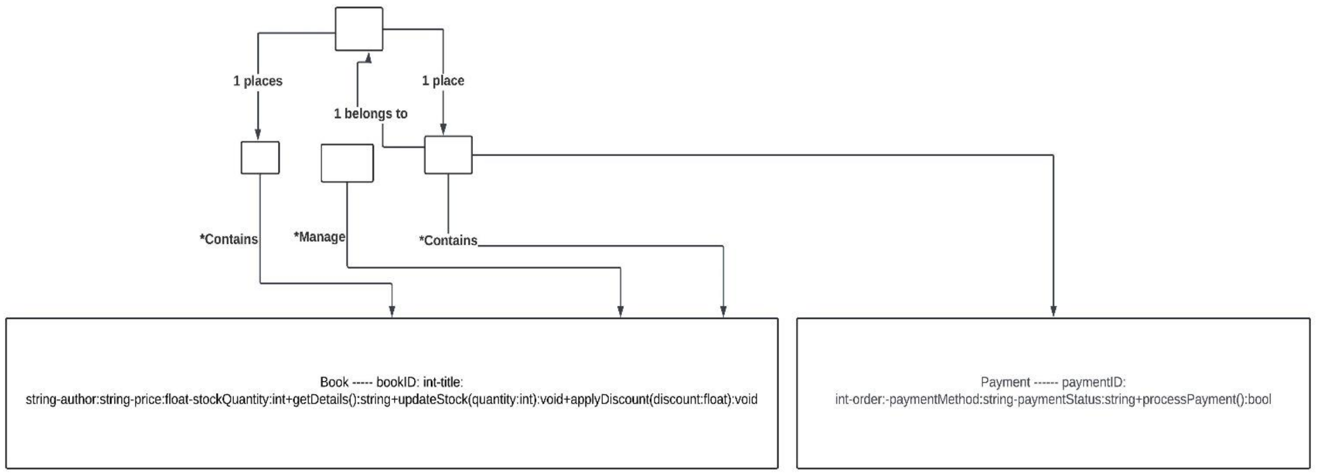
The applyDiscount(percentage: float) and calculateItemTotal() methods can be extended with value restrictions, which would define a minimum permitted price of zero following the application of discounts. Furthermore, to make sure that no negative or excessively high numbers are produced as a result of calculation mistakes, the calculateTotal() function in the Cart and Order classes should check for realistic totals.

Part 2

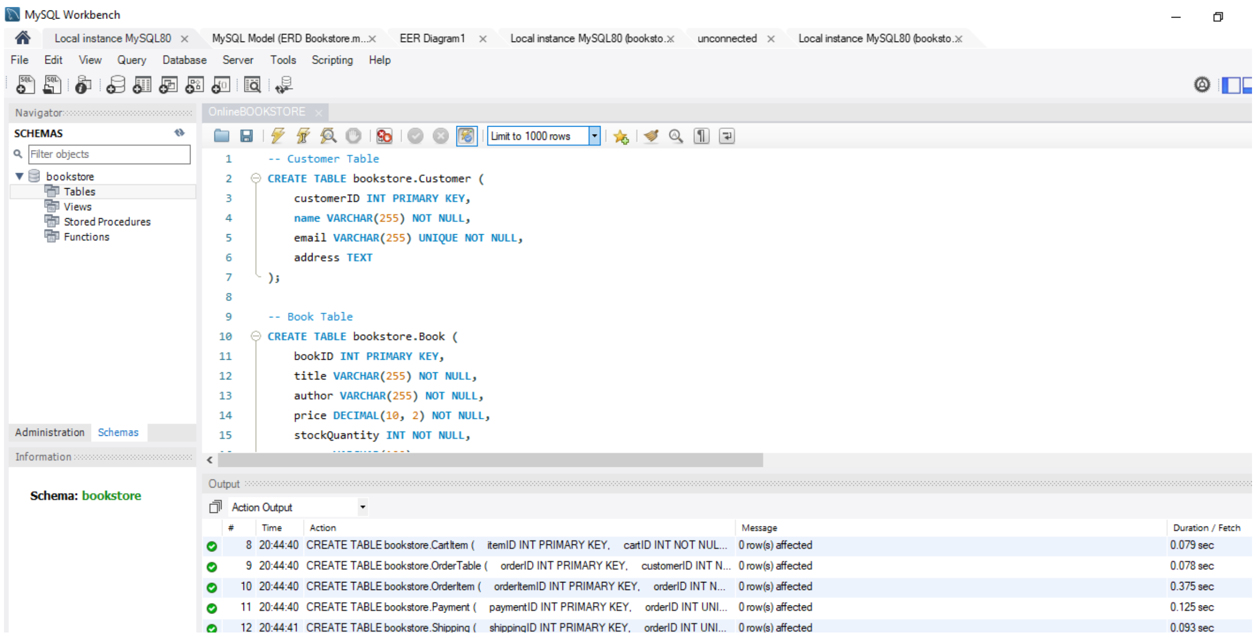
Entity Relationship Diagram



UML Class Diagram


1 places
1 place
1 belongs to
*Contains
*Manage
*Contains
Book ----- bookID: int-title:
string-author:string-price:float-stockQuantity:int+getDetails():string+updateStock(quantity:int):void+applyDiscount(discount:float):void
Payment ------ paymentID:
int-order:-paymentMethod:string-paymentStatus:string+processPayment():bool

Part 3



Schema: bookstore

```
-- Customer Table
CREATE TABLE bookstore.Customer (
    customerID INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    address TEXT
);

-- Book Table
CREATE TABLE bookstore.Book (
    bookID INT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    stockQuantity INT NOT NULL,
```

```
#   Time      Action                                                                        Message            Duration / Fetch
8   20:44:40  CREATE TABLE bookstore.CartItem (   itemID INT PRIMARY KEY,   cartID INT NOT NUL...     0 row(s) affected  0.079 sec
9   20:44:40  CREATE TABLE bookstore.OrderTable (   orderID INT PRIMARY KEY,   customerID INT N...  0 row(s) affected  0.078 sec
10  20:44:40  CREATE TABLE bookstore.OrderItem (   orderItemID INT PRIMARY KEY,   orderID INT N...  0 row(s) affected  0.375 sec
11  20:44:40  CREATE TABLE bookstore.Payment (   paymentID INT PRIMARY KEY,   orderID INT UNI...    0 row(s) affected  0.125 sec
12  20:44:41  CREATE TABLE bookstore.Shipping (   shippingID INT PRIMARY KEY,   orderID INT UNI...  0 row(s) affected  0.093 sec
```

Customer Table

1NF (No repeating groups), 2NF (Each non-key attribute depends on the entire primary key), and 3NF (No transitive dependencies)

Book Table

3NF (The Book table doesn't have partial or transitive dependencies)

Cart Table

3NF (Each cart is associated with a unique customer)

CartItem Table

1NF (Each item is atomic), 2NF, and 3NF (Associates books with a specific cart)

Order Table

3NF (Each order is linked to a single customer)

OrderItem Table

3NF (Avoids redundancy by linking each order item to a specific book and order)

Payment Table

3NF (Payment data is kept separate and linked to an order)

Shipping Table

3NF (Separates shipping information from order details for modularity)

Explanation of Normalization

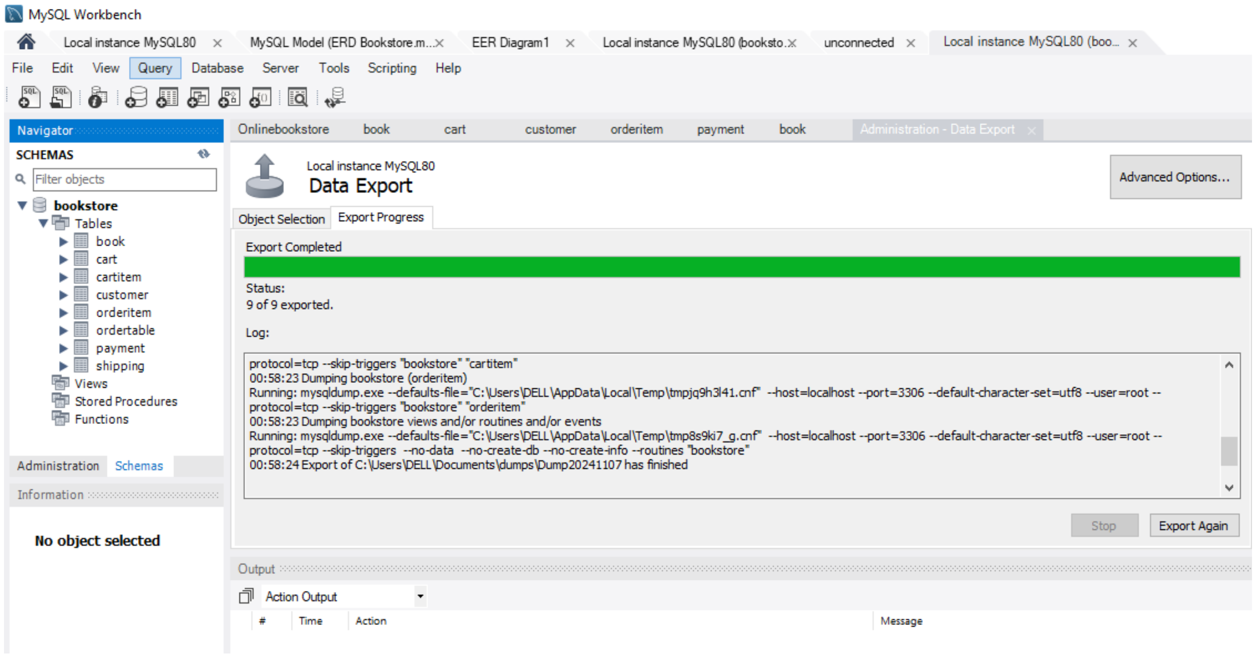
1. 1NF: All tables have atomic attributes with no repeating groups.
2. 2NF: There are no partial dependencies since each non-key attribute depends on the whole primary key.
3. 3NF: There are no transitive dependencies. For example:

Customer has no dependencies other than customerID.

Book details are isolated, with unique identifiers like ISBN.

Order information is separated from Payment and Shipping, ensuring modular data handling and reduced redundancy.

Part 4

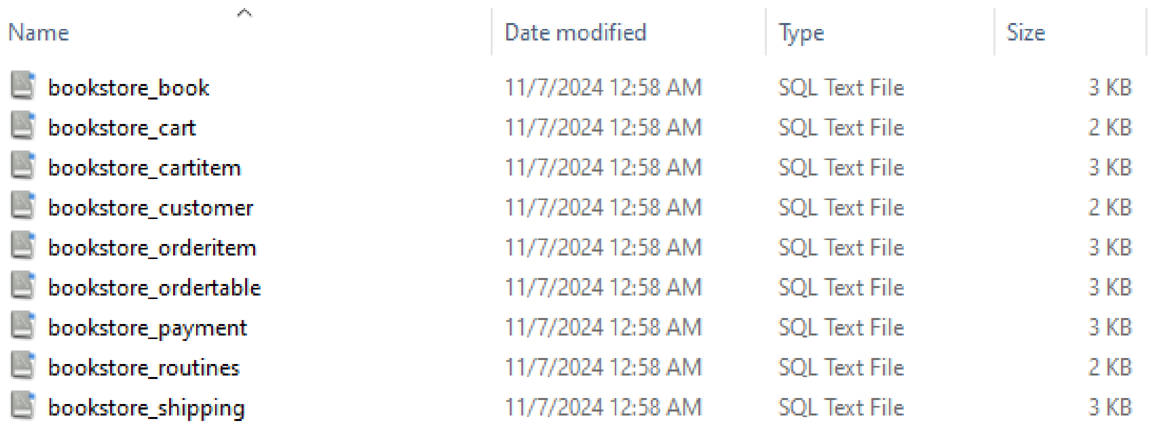


Export Completed

Status:
9 of 9 exported.

Log:

```
protocol=tcp --skip-triggers "bookstore" "cartitem"
00:58:23 Dumping bookstore (orderitem)
Running: mysqldump.exe --defaults-file="C:\Users\DELL\AppData\Local\Temp\tmpjq9h3l41.cnf"  --host=localhost --port=3306 --default-character-set=utf8 --user=root --
protocol=tcp --skip-triggers "bookstore" "orderitem"
00:58:23 Dumping bookstore views and/or routines and/or events
Running: mysqldump.exe --defaults-file="C:\Users\DELL\AppData\Local\Temp\tmp8s9ki7_g.cnf"  --host=localhost --port=3306 --default-character-set=utf8 --user=root --
protocol=tcp --skip-triggers  --no-data  --no-create-db --no-create-info --routines "bookstore"
00:58:24 Export of C:\Users\DELL\Documents\dumps\Dump20241107 has finished
```



Name	Date modified	Type	Size
bookstore_book	11/7/2024 12:58 AM	SQL Text File	3 KB
bookstore_cart	11/7/2024 12:58 AM	SQL Text File	2 KB
bookstore_cartitem	11/7/2024 12:58 AM	SQL Text File	3 KB
bookstore_customer	11/7/2024 12:58 AM	SQL Text File	2 KB
bookstore_orderitem	11/7/2024 12:58 AM	SQL Text File	3 KB
bookstore_ordertable	11/7/2024 12:58 AM	SQL Text File	3 KB
bookstore_payment	11/7/2024 12:58 AM	SQL Text File	3 KB
bookstore_routines	11/7/2024 12:58 AM	SQL Text File	2 KB
bookstore_shipping	11/7/2024 12:58 AM	SQL Text File	3 KB