Problem:

Suppose we're building a simple online bookstore. We need to design a relational model for managing books, authors, customers, and orders.

Entities:

1. **Books**: Each book has a title, author, price, and quantity available.
2. **Authors**: Authors have a name and possibly other details.
3. **Customers**: Customers have a name, email, and address.
4. **Orders**: Each order is made by a customer and contains one or more books.

Relational Model:

- **Books** (book_id, title, author_id, price)
- **Authors** (author_id, name)
- **Customers** (customer_id, name, email, address)
- **Orders** (order_id, customer_id, order_date)
- **OrderDetails** (order_id, book_id, quantity)

Deliverable 1: Create the above relational model in SQL Server Database with the scripts needed in sql format.

Deliverable 2: Create a script to insert 10 records in each table created in Deliverable 2

Deliverable 3**:** Formulate the queries needed to retrieve the information from the previous model**:**

1. **Get all books by a specific author**
2. **Get the total number of books sold**
3. **Get the total revenue generated from orders**
4. **Get the top 5 best-selling books**
5. **Get the customers who have spent the most**

API Creation Question:

Instructions:

- Use Node.js with Express.js to create a RESTful API.
- The API should have two endpoints to interact with the database:
    1. Endpoint to retrieve information about orders.
    2. Endpoint to retrieve information about customers.
- Ensure error handling and appropriate status codes for responses.
- Document the API using appropriate comments.

Endpoint 1: Retrieve Orders Information

- **URL: /api/orders**
- **Method: GET**
- **Description:** Retrieve information about all orders in the database including book details.
- **Response:**
    - Status Code: **200 OK**
    - Body: JSON array containing information about each order, including order id, book title, author, price, and quantity.

Endpoint 2: Retrieve Customers Information

- **URL: /api/customers**
- **Method: GET**
- **Description:** Retrieve information about all customers in the database.
- **Response:**
    - Status Code: **200 OK**
    - Body: JSON array containing information about each customer, including name, email, and address.

<u>Deliverable 4</u>**:**

- Implement the API using Node.js with Express.js.
- Test the API using tools like Postman or curl.

- Ensure the API endpoints are functional and return the expected data from the database.
- Provide appropriate error handling and status codes in responses.
- Submit your code as a single Node.js file or a directory containing relevant files.

Question: Create a User Interface to Display API Data

Instructions:

- Create a simple web-based user interface using HTML, CSS, and JavaScript.
- The interface should consist of two modules, each displaying data retrieved from one of the API endpoints you implemented earlier.
- Use React.js as frontend framework
- Ensure the user interface is responsive and visually appealing.
- Provide appropriate error handling if the API endpoints fail to respond.

Module 1: Orders Information

- Display information about all orders retrieved from the **/api/orders** endpoint.
- Render the orders data in a table format with columns for order id, book title, author, price, and quantity.

Module 2: Customers Information

- Display information about all customers retrieved from the **/api/customers** endpoint.
- Render the customer data in a table format with columns for name, email, and address.

Bonus (Optional):

- Add pagination or infinite scrolling functionality to handle large datasets efficiently.
- Implement filtering or search functionality to allow users to search for specific books or customers by name, title, or other attributes.

Deliverable 5**:**

- Implement the user interface using the provided instructions.
- Test the interface to ensure it correctly displays data from the API endpoints.
- Provide appropriate error handling if the API requests fail.
- Submit your code as a single HTML file or a directory containing relevant HTML, CSS, and JavaScript files.