Laboratory work #10

Please write SQL queries for following tasks and save as .sql file.

Database Structure: Online Bookstore

Tables:

1. Books
   - `book_id` (Primary Key, Integer)
   - `title` (Varchar)
   - `author` (Varchar)
   - `price` (Decimal)
   - `quantity` (Integer)

2. Orders
   - `order_id` (Primary Key, Integer)
   - `book_id` (Foreign Key, Integer)
   - `customer_id` (Integer)
   - `order_date` (Date)
   - `quantity` (Integer)

3. Customers
   - `customer_id` (Primary Key, Integer)
   - `name` (Varchar)
   - `email` (Varchar)

Sample Data:

Books

book_id	title	author	price	quantity
1	Database 101	A. Smith	40.00	10
2	Learn SQL	B. Johnson	35.00	15
3	Advanced DB	C. Lee	50.00	5

Customers

customer_id	name	email
101	John Doe	johndoe@example.com
102	Jane Doe	janedoe@example.com

Tasks:

1. Transaction for Placing an Order
   - Scenario: A customer places an order for a book. This should decrease the quantity of the book in the `Books` table and create a new record in the `Orders` table.
   - Task:
     - Start a transaction.
     - Insert an order into the `Orders` table for `customer_id` 101 ordering 2 quantities of `book_id` 1.
     - Update the `Books` table by reducing the quantity of `book_id` 1 by 2.
     - Commit the transaction.
     - Expected Outcome: The `Orders` table has a new record, and the `Books` table shows 8 as the new quantity for `book_id` 1.

2. Transaction with Rollback
   - Scenario: A customer attempts to order more books than are in stock, which should not be allowed.
   - Task:
     - Start a transaction.
     - Attempt to insert an order into the `Orders` table for `customer_id` 102 ordering 10 quantities of `book_id` 3.
     - Check if the quantity in `Books` is sufficient. If not, rollback the transaction.
     - Expected Outcome: No change in the `Orders` or `Books` tables due to the rollback.

3. Isolation Level Demonstration
   - Scenario: Show the effect of different isolation levels in concurrent transactions.
   - Task:
     - In one session, start a transaction at READ COMMITTED isolation level, and update the price of a book in the `Books` table.

- In a second session, start another transaction at the same isolation level and read the price of the same book.
- Commit the first transaction and re-read the price in the second session.
- Expected Outcome: The second session reads the updated price after the first transaction commits, demonstrating the READ COMMITTED isolation level.

4. Durability Check
- Scenario: Ensure that the changes made by a transaction are permanent.
- Task:
  - Perform a transaction where you update a customer's email in the `Customers` table.
  - Commit the transaction.
  - Restart the database server.
  - Check the `Customers` table for the update.
  - Expected Outcome: The new email address persists even after a database restart, demonstrating durability.