**BOOK SHOP**

**Requirements:**

Write a report which will have the following structure:

- Introduction to the system

- ER diagram

- Explanation of why the structure follows normal forms

- Explanation and coding part of each item

**1. Introduction to the system:**

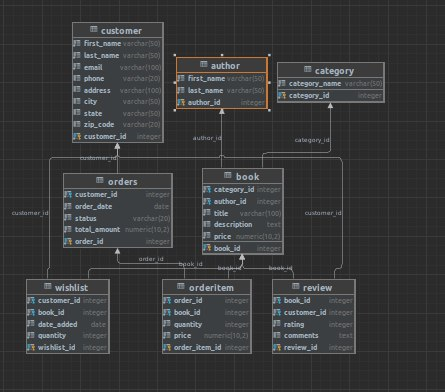
The book shop system is an online store where you can buy books

easily. You can search for books, see all the details, and pay for them

securely online. The system is also great for the admin because they

can manage all the books, orders, and reports efficiently

**2. ERD:**



**3. Explanation of why the structure follows normal forms (1NF, 2NF,**

**3NF):**

✔️ The database schema follows the rules of normalization to reduce data

redundancy and maintain data integrity. The tables are structured in

such a way that they follow the 1st, 2nd, and 3rd normal forms.

✔️ 1NF (First Normal Form): The database schema follows the 1st normal

form, which ensures that each column in the table has atomic values.

Each table has a primary key that uniquely identifies each row in the

table. For example, the Customers table has a primary key

'customer\_id' that uniquely identifies each customer.

✔️ 2NF (Second Normal Form): The database schema follows the 2nd

normal form, which ensures that all non-key attributes in a table are

dependent on the primary key. The Orderitem table, for instance, has a

composite primary key 'order\_id' and 'book\_id' that uniquely identifies

each row in the table. The non-key attributes in the table, such as

'quantity' and 'price,' depend on the primary keys.

✔️ 3NF (Third Normal Form): The database schema follows the 3rd normal

form, which ensures that all non-key attributes in a table are

independent of each other. The database schema avoids transitive

dependencies, and each table is designed to store only one type of

information. For instance, the Category table only stores information

related to book categories and does not contain information related to

Orders or Customers**.**

**4. Explanation and coding part of each item:**

**Tables Creation:**

➢ **Books:** This table contains information about the books, such as the

name, author, price, and category\_id.

➢ **Category:** This table contains information about the book categories,

such as the category\_id and category\_name.

➢ **Customers:** This table contains information about the customers, such

as their name, email, password, phone number, address, city and etc.

➢ **Orderitem:** This table contains information about the items in the

orders, such as the order\_id, book\_id, quantity, and price.

➢ **Orders:** This table contains information about the orders, such as the

order\_id, customer\_id, order\_date, and order\_number.

➢ **Reviews:** This table contains information about the reviews, such as

the book\_id, customer\_id, and review\_text.

➢ **Wishlist:** This table contains information about the books that the

customers have added to their wishlist, such as the book\_id and

customer\_id.

**The following procedures were created for this project:**

1. **get\_book\_group\_totals():** The procedure is used to generate a report

of the total number of books and the total price for each author within

each category.

2**. update\_book\_price():** The code defines a PostgreSQL stored

procedure named update\_book\_price that takes two parameters: id

of type integer and new\_price of type numeric with precision 10 and

scale 2. The procedure updates the price of a book with the given ID in

the book table to the new price. If the book is found and updated

successfully, a notice message is raised with the number of books

updated. If the book with the given ID is not found, an exception is

raised.

**The following functions were created for this project:**

1. **count\_records():** this function counts all records from a table which

user inputs.

2. **show\_row\_count():** this code sets up a trigger that counts the

number of rows in the customer table before each new row is inserted.

This can be useful for monitoring the growth of the table over time.

3. **validate\_title():** The procedure uses two nested loops to iterate

through each distinct category and author in the "book" table. For each

category and author, it retrieves the total count of books and the total

price for that category and author using the "COUNT" and "SUM"

functions. It then uses the "RAISE NOTICE" statement to print a

message to the console with the category ID, author ID, book count,

and total price.

**The following trigger was created for this project:**

1. **update\_book\_rating\_trigger:** This trigger is designed to update the

rating of a book whenever a new review is added to the Reviews table.

It uses a stored procedure to calculate the average rating for the book

based on all the reviews and updates the Books table with the new

rating.