BOOK SHOP

Requirements:

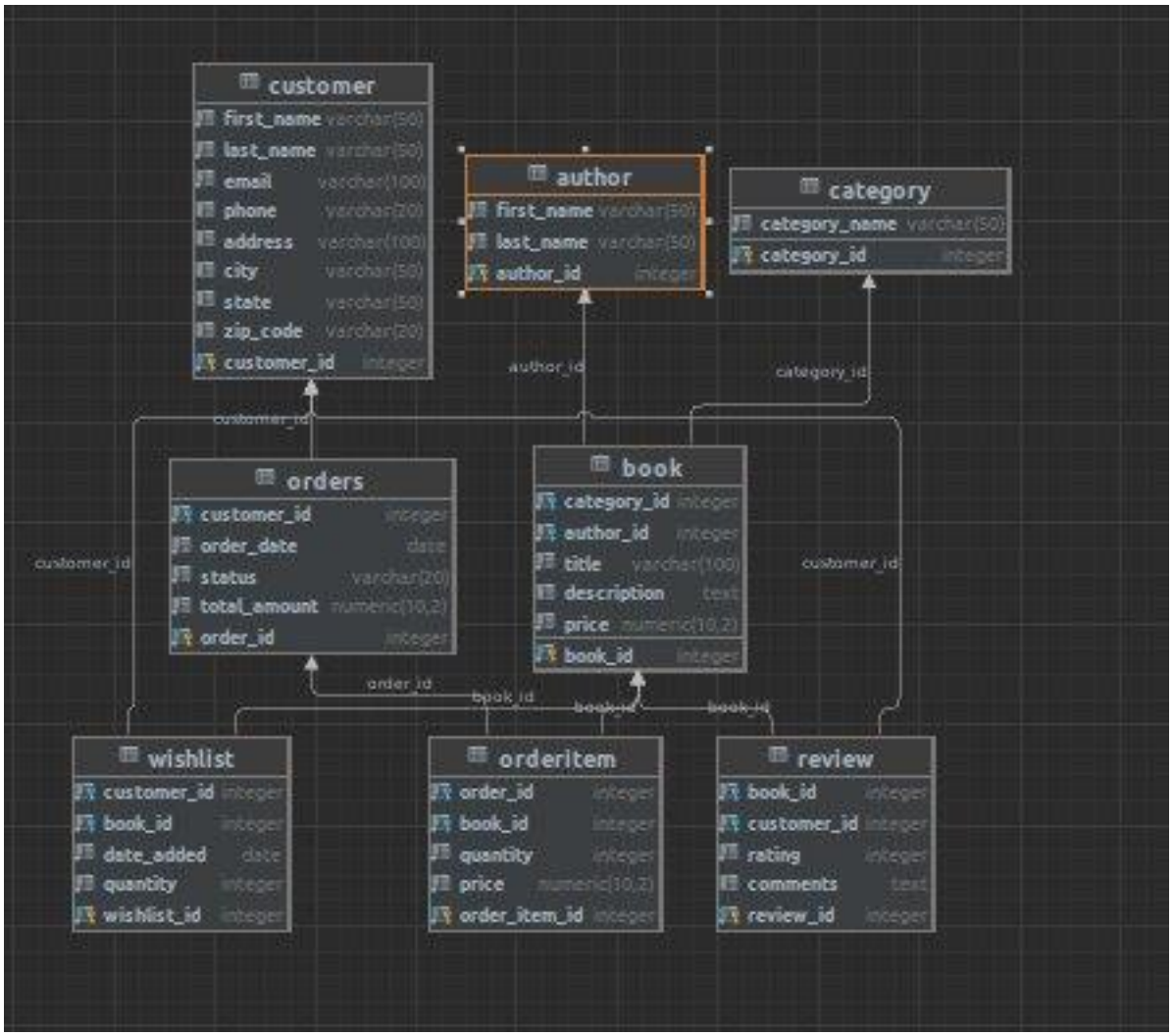
Write a report which will have the following structure:

- 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:


customer
first_name varchar(50)
last_name varchar(50)
email varchar(100)
phone varchar(20)
address varchar(100)
city varchar(50)
state varchar(50)
zip_code varchar(20)
customer_id integer
author
first_name varchar(50)
last_name varchar(50)
author_id integer
category
category_name varchar(50)
category_id integer
author_id
category_id
customer_id
orders
customer_id integer
order_date date
status varchar(20)
total_amount numeric(10,2)
order_id integer
book
category_id integer
author_id integer
title varchar(100)
description text
price numeric(10,2)
book_id integer
customer_id
customer_id
order_id
book_id
book_id
book_id
wishlist
customer_id integer
book_id integer
date_added date
quantity integer
wishlist_id integer
orderitem
order_id integer
book_id integer
quantity integer
price numeric(10,2)
order_item_id integer
review
book_id integer
customer_id integer
rating integer
comments text
review_id integer

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.