Embedded SQL Queries in BookHaven Application

Overview

This README provides a summary of the embedded SQL queries used in the BookHaven application. These queries interact with the MySQL database to perform various operations such as user authentication, registration, cart management, and book browsing.

Queries

User Authentication and Registration

1. **Local Strategy Authentication**
    - **Purpose:** Authenticates users using local strategy (email and passkey).
    - **Query:** `SELECT * FROM Customer WHERE email=? AND passkey=?`
2. **User Registration**
    - **Purpose:** Registers a new user into the database.
    - **Query:**

```
INSERT INTO customer (first_name, last_name, pincode, address, contact, date_of_birth, email, passkey, amount, owned_books)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

Admin Operations

1. **Admin Login**
    - **Purpose:** Allows an admin to log in.
    - **Query:** `SELECT * FROM Admin WHERE username=? AND passkey=?`
2. **Inventory and Customer Management**
    - **Purpose:** Retrieves all customers and books from the database.
    - **Query:**

```
SELECT * FROM customer;
SELECT * FROM book;
```

Book Operations

1. **Browse Books**
    - **Purpose:** Retrieves all books from the database for browsing.
    - **Query:** `SELECT * FROM book`

Cart Management

1. **Add Book to Cart**
    - **Purpose:** Adds a book to the user's cart.
    - **Query:** `CALL insert_or_update_cart_item (?, ?)`
    - **Note:** This is a stored procedure to insert or update cart items.
2. **View Cart**
    - **Purpose:** Retrieves all items in the user's cart.
    - **Query:**

```
SELECT book.book_name, book.author_name, book.genre, book.price, cart_items.count
FROM cart_items
JOIN book ON cart_items.book_id = book.book_id
WHERE cart_items.cart_id = ?
```

Trigger:

1. **Maintaining Book Stock on Order**

```
DELIMITER $$
CREATE TRIGGER reduce_stock_on_order
BEFORE INSERT ON `order`
FOR EACH ROW
BEGIN
DECLARE current_stock INT;

SELECT stock INTO current_stock FROM book WHERE book_id = NEW.book_id;

IF current_stock < NEW.quantity THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Insufficient book stock to fulfill order.';
END IF;

UPDATE book
SET stock = stock - NEW.quantity
WHERE book_id = NEW.book_id;
END $$
DELIMITER ;
```

2. **Ensuring User is 13+**

```
DELIMITER $$
CREATE TRIGGER check_age_before_insert_update
BEFORE INSERT ON customer
FOR EACH ROW
BEGIN
    IF DATEDIFF(NOW(), NEW.date_of_birth) < 4745 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Age must be 13 or older';
    END IF;
END$$
DELIMITER ;
```

3. **Ensuring a book added to cart is not *out-of-stock* or *inappropriately-rated***

```
DELIMITER $$
DROP TRIGGER IF EXISTS check_book_before_insert_cart$$
CREATE TRIGGER check_book_before_insert_cart
BEFORE INSERT ON Cart_Items
FOR EACH ROW
BEGIN
    DECLARE is_eligible INT;
    DECLARE current_stock INT;

    -- Check if the book is eligible to be placed in the cart
    SELECT IFNULL(
        CASE
            WHEN b.age_rating = 'A' AND cu.date_of_birth <= DATE_SUB(CURRENT_DATE(), INTERVAL 18 YEAR) THEN 1
            WHEN b.age_rating != 'A' THEN 1
            ELSE 0
        END, 1)
    INTO is_eligible
    FROM Book b
    INNER JOIN Cart c ON c.cart_id = NEW.cart_id
    INNER JOIN Customer cu ON cu.customer_id = c.customer_id
    WHERE b.book_id = NEW.book_id;

    IF is_eligible = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Only customers aged 18 or older can add A-rated books to their cart.';
    END IF;

    -- Check if the book is in stock
    SELECT stock INTO current_stock FROM Book WHERE book_id = NEW.book_id;

    IF current_stock < 1 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The book is out of stock.';
    END IF;
END$$
DELIMITER ;
```

4. **Handle dupliate cart_item entries**

```
DELIMITER $$

DROP TRIGGER IF EXISTS update_cart_item_count$$

CREATE PROCEDURE insert_or_update_cart_item(
    IN cart_id_param INT,
    IN book_id_param INT
)
BEGIN
    DECLARE existing_count INT;

    -- Check if there's already an entry for the same cart and book
    SELECT `count` INTO existing_count
    FROM Cart_Items
    WHERE cart_id = cart_id_param AND book_id = book_id_param;

    -- If there's an existing entry, update its count
    IF existing_count IS NOT NULL THEN
        UPDATE Cart_Items
        SET `count` = `count` + 1
        WHERE cart_id = cart_id_param AND book_id = book_id_param;
    ELSE
        -- Otherwise, insert a new row
        INSERT INTO Cart_Items (cart_id, book_id, `count`)
        VALUES (cart_id_param, book_id_param, 1);
    END IF;
END$$

DELIMITER ;
```

Conclusion

This document provides an overview of the embedded SQL queries used in the BookHaven application. These queries facilitate various functionalities such as user authentication, registration, admin operations, book browsing, and cart management. Many more queries have been used and will be used further along the development of the application.