Database Management Systems - I Lab

CSE-2211 Project

Project Title

Online Book Store

Project Area Major

E-Commerce

Project Area Minor

Commercial Book Store Management

Submitted By

Md. Mahmudul Hasan (Roll - 10)

Submitted To

Dr. Md. Mustafizur Rahman

Dr. Muhammad Ibrahim

Submitted On

18 November 2023

Introduction

1.1 About the Project

In a world touched by the enchantment of technology, our online bookstore beckons as a warm haven for bibliophiles. Picture a digital realm where the simplicity of ordering, searching, and purchasing books is seamlessly intertwined with the joy of literary exploration. Here, our exclusive focus on the captivating world of books creates a curated environment, a sanctuary that cherishes the written word in all its delightful forms. Beyond being a mere marketplace, we aspire to cultivate a vibrant community of readers, united by their shared passion for books.

Our platform doesn't just facilitate transactions; it empowers a personalized journey for each reader. With the ease of ordering at your fingertips, exploring our virtual shelves becomes an adventure, enhanced by the availability of enchanting ebooks that open new dimensions of reading pleasure.

The heart of our mission lies in curating a collection that transcends time and genres. From the enduring classics to the pulse of contemporary literature, every book is carefully chosen to inspire, educate, and entertain. We believe in the transformative power of literature to shape minds and broaden perspectives. In this harmonious blend of tradition and innovation, we pay homage to the timeless allure of books while embracing the limitless possibilities of the digital age.

Our online bookstore is more than a destination; it's an invitation to a world where community thrives, connections are forged through shared tales, and the love for books knows no bounds. So, we tried to make a digital platfrom, where the magic of literature unfolds, stories come alive, and imaginations take flight, and make every bookworm’s life a bit easier to discover new books and order them from our platform at any time from any where in the world.

1.2 Motivation

The motivation behind embarking on the Online Bookstore Database project was deeply rooted in a passion for literature and a vision to transform the traditional book-buying experience into a seamless, digital journey. As an ardent reader, the realization dawned that in an era dominated by digital innovation, the world of books could benefit immensely from a platform that marries the timeless charm of literature with the convenience of modern technology.

The motivation stemmed from a desire to create more than just an online marketplace. It was about crafting a virtual haven, a dedicated space where bibliophiles could not only explore and discover an extensive array of literary works but also order them effortlessly from any corner of the world. The aim was to build a community united by their shared love for books, transcending geographical boundaries and time zones. The inspiration further crystallized in the recognition of the challenges faced by readers in navigating the vast literary landscape. The project sought to address these challenges by offering a specialized and curated environment focused exclusively on books. It was a call to bridge the gap between readers and the vast world of literature.

1.3 Key Features

- **Extensive Book Catalog:**
  A wide-ranging collection of books covering various genres, including fiction, non-fiction, academic, and specialized categories.

- **Search and Filter Options**
  Robust search functionality to help users find specific books quickly. Filters based on genres, authors, language, and other criteria for a refined search experience.

- **User Accounts**
  User registration and account management for personalized experiences. Features like order history, wish lists, and recommendations based on past purchases.

- **Admin Accounts**
  Specialized administrative accounts with the capability to manage and oversee the entire online bookstore platform. Admins can monitor user activity, manage inventory, analyze data, and make necessary updates to the system.

- **Product Pages**
  Detailed product pages for each book, featuring descriptions, author information, pricing, and user reviews.

- **E-commerce Functionality**
  Seamless online ordering system, allowing users to add books to a shopping cart and proceed through a secure checkout process.

- **E-book Availability**
  Option to purchase and download electronic versions (e-books) for supported titles.

- **Promotions and Discounts**
  Regular promotions, discounts, and special offers to incentivize purchases.

- **Secure Payment Options**
  Multiple secure payment options, including cash on delivery and credit/debit cards, mobile wallets, and other online payment methods.

1.4 Tools & Technologies used

- **Database Management System (DBMS):** MySQL
- **Database Design and Visualization:** MySQL Workbench
- **Integrated Development Environment (IDE):** IntelliJ IDEA
- **Programming Language:** Java

Database Design & Implementation

2.1 About the Database

2.1.1 Brief Discussion

In crafting our Online Bookstore database, meticulous attention has been dedicated to creating a comprehensive structure that seamlessly integrates key entities such as authors, publishers, genres, books, users, orders, and administrators. Employing primary and foreign key constraints, along with normalization principles, ensures data integrity and efficient organization. This database is designed to facilitate a myriad of user queries, offering dynamic searches based on writers, genres, and publications. It enables users to explore the latest books, those with special offers, and e-book availability, while also providing insights into stock levels and bestsellers. With features encompassing order tracking, detailed book information, and advanced analytics, our database serves as the backbone of an immersive and user-friendly online bookstore experience, catering to diverse user needs and preferences.

2.1.2 Detailed Discussion

In the development of our Online Bookstore, a robust and well-organized database has been carefully designed, encompassing key entities to ensure a comprehensive and user-centric experience.

The Author entity celebrates the creative minds behind literary masterpieces, emphasizing the importance of recognizing and categorizing their works. Publishers, highlighted in the Publisher entity, play a crucial role in bringing these stories to the world. The Book entity serves as a centralized hub for book details, enabling users to search, explore, and order books seamlessly. Genres, systematically categorized in the Genre entity, enhance the browsing experience. The User entity facilitates a personalized interactive space for book enthusiasts, while the Admin entity empowers administrators to manage orders, ensure payment accuracy, and oversee overall sales and revenue.

Users can effortlessly search for books based on the writer's name, genre, or publication details, ensuring a tailored exploration of the vast literary collection. The query to display the latest books caters to the user's desire for up-to-date literary discoveries, while the query showcasing books with special offers adds an exciting dimension to the shopping experience.

For users keen on digital reads, the query identifying books with e-book availability becomes a valuable tool. The stock-related queries offer real-time insights into book availability, empowering users with information on the number of copies in stock and aiding in decision-making.

In the realm of analytics, the queries provide a diverse range of insights. Users can explore bestsellers on weekly, monthly, and yearly bases, view book prices, and even order books based on their prices—whether from low to high or vice versa.

For administrators, the database supports comprehensive order tracking, ensuring efficient management. Admins can monitor order history, track shipment and payment statuses, and gain insights into the total sales for each book. Additionally, the queries facilitate a deeper understanding of customer behavior, allowing admins to identify top spenders, best-selling authors, and popular genres.

The database's analytic capabilities extend to financial aspects, with queries revealing total sales, revenue generated by each author, and the percentage of orders paid through online services. Furthermore, administrators can identify customer segments who have not made any purchases, allowing for targeted engagement strategies.

With a focus on market trends, the database offers insights into customer preferences, such as the top genres with the highest total sales and customers who have purchased books in more than one genre. Additionally, the database supports queries related to average quantity of books ordered per customer, books ordered more than once, and even books frequently bought together—providing valuable information for strategic inventory management.

2.2 Database Design

The genesis of our database project finds its roots in the pursuit of creating an immersive and efficient online bookstore. It all started with the **Author** entity, a homage to the creative minds behind literary masterpieces. Authors, each with a unique story to tell, became the cornerstone of our database, emphasizing the need to recognize and categorize their works. This led to the inception of the **Publisher** entity, acknowledging the crucial role publishers play in bringing these stories to the world.

As the narrative unfolded, it became evident that readers needed a way to navigate the vast literary landscape. Hence, the **Genre** entity took shape, offering a systematic approach to categorizing books and aiding in the discovery of new and diverse narratives. However, a bookstore is not just about individual works; it's about the collective experience of literature. Thus, the **Book** entity emerged, serving as the central repository for all details related to a literary work, from its title and format to its availability and pricing.

The **User** entity became a crucial addition, recognizing that a bookstore is not merely a collection of books but an interactive space for enthusiasts. Users could now register, explore the curated literary collection, and place orders seamlessly. In the continuum of our narrative, the **Order** entity seamlessly integrates into this interactive landscape. It becomes the vital link between users and the books they desire, capturing the essence of transactions and order details. Lastly, Administrators, with the **Admin** entity, gained the tools to manage and oversee the entire operation, ensuring a smooth and secure digital environment.

We found a traditional relationship between authors and books – it's like a teamwork where one author can contribute to many books, and one book can involve contributions from different authors. This kind of connection is called a many-to-many relationship. To manage this, we introduced the **AuthorBook** table. This table acts like a bridge, telling us which author is involved with which book. It's a way of capturing the teamwork in the literary world. We made it so that authors can

choose to be part of this connection, but for books, it's a must – every book needs to have some authors. So, the AuthorBook table is like a key player in showing the creative partnerships between authors and books in our database.

Soon we figured out a very important relationship between books and publishers. In our design, we see it as a many-to-one relationship, signifying that while one publisher can be associated with multiple books, each book is uniquely linked to a single publisher. This relationship is crucial for understanding the collaborative effort in bringing books to readers. To navigate this connection, we introduced a special identifier known as the publisher ID within the book table. This ID serves as a foreign key, referencing the primary key of the publisher table. Essentially, it acts like a code that points to the specific publisher responsible for a given book. Importantly, we established that every book must be linked to a publisher, making it a mandatory part of the book's identity, while a publisher has the flexibility to have numerous associated books. This meticulous integration of publisher IDs enhances the clarity of our database, allowing us to seamlessly trace the origin of each book back to its publishing source.

Furthermore, we've identified a significant association between genres and books. This relationship is characterized as a many-to-many relationship, signifying that a single book may fall under various genres, and conversely, each genre may encompass multiple books. To systematically manage this relationship, we introduced the **GenreBook** table. This table serves as a structured record, delineating the specific genres to which each book belongs. Effectively, it functions as a tool to denote, "This particular book is categorized under these genres." We've established the flexibility for a book to be associated with multiple genres and for a genre to encompass numerous books.

The relationship between the Order and Book entities unfolds as a many-to-many connection, reflecting the intricate nature of user transactions. In each order, multiple books are listed, forming a complex web of associations. To streamline this relationship and ensure accurate recording, the **OrderBook** table acts as a bridge, capturing which specific books are included in each order. This allows users to delve into their order history, discerning the assortment of books chosen in each transaction. The introduction of the OrderBook table significantly enriches the depth of our database, enabling a nuanced exploration of the dynamic connections between orders and books within our online bookstore.

Lastly, the relationship between the Order and User entities takes the form of a many-to-one relationship, reflecting the association between users and their

respective orders. In this structure, each order is linked to a specific user, highlighting the personalized nature of transactions within the online bookstore. To seamlessly integrate this relationship, a foreign key is introduced within the Order table. This key serves as a reference to the primary key of the User table, establishing a clear connection between orders and the users who initiate them. This integration ensures that each order is attributed to a specific user, facilitating efficient order tracking and providing a personalized experience for users within our online bookstore system.

In essence, each entity and relationship serves a distinct purpose, contributing to the holistic experience of our online bookstore. Authors and Publishers give life to stories, Genres offer a roadmap for exploration, Books house the narratives, Users drive engagement, and Admins ensure the seamless operation of this literary ecosystem. The database, therefore, stands not just as a collection of tables but as a vibrant narrative woven together by the shared love for books and literature.

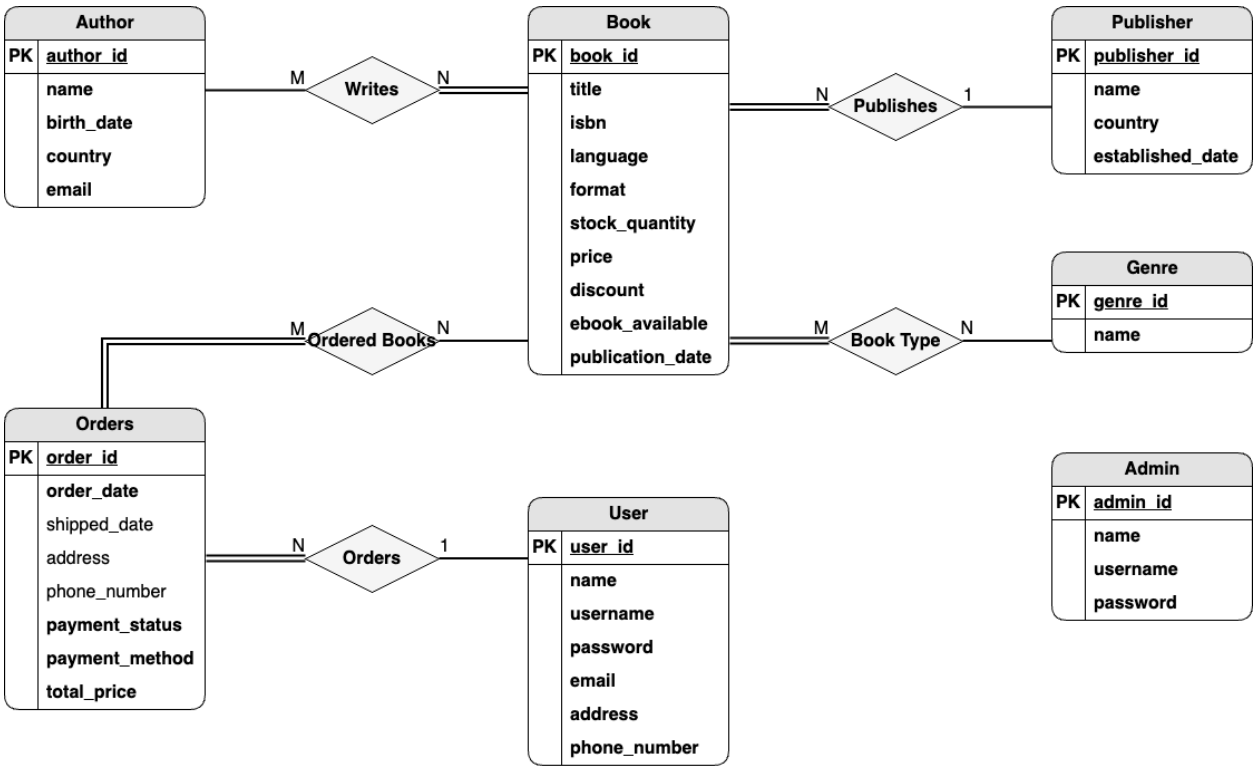

Figure: ER Diagram of BookStore Diagram

The list of all the tables of the database is given below.

Author *(author_id, name, birth_date, country, email)*
Publisher *(publisher_id, name, country, established_date)*
Genre *(genre_id, name)*
Book *(book_id, title, publisher_id, isbn, number_of_pages, language, format, stock_quantity, price, discount, ebook_available, publication_date)*
User *(user_id, name, email, username, password, address, phone_number)*
Orders *(order_id, user_id, order_date, shipped_date, address, phone_number, payment_status, payment_method, total_price)*
AuthorBook *(author_id, book_id)*
GenreBook *(book_id, genre_id)*
OrderBook *(order_id, book_id, quantity)*
Admin *(admin_id, name, username, password)*

Author

S/N	Attribute	Data Type	Constraint(If any)
1	author_id	INT	Primary key
2	name	VARCHAR	Not null
3	birth_date	DATE	Not null
4	country	VARCHAR	Not null
5	email	VARCHAR	Not null
This Table, anchored by the “author_id” primary key, stores crucial information such as author names, birth dates, countries, and email addresses. This data serves to enrich user experiences by providing insights into authors, enabling effective categorization, and forming essential connections between authors and their literary works within the Online Bookstore Management System.			

Publisher

S/N	Attribute	Data Type	Constraint(If any)
1	publisher_id	INT	Primary key
2	name	VARCHAR	Not null
3	country	VARCHAR	Not null
4	established_date	Date	Not null
This table stores essential information about book publishers, including their names, countries, and establishment dates. It serves as a central repository for efficiently identifying publishers, supporting book metadata, and facilitating queries and reporting related to publishers within an online bookstore database. The primary key constraint ensures data integrity by uniquely identifying each publisher.			

Genre

S/N	Attribute	Data Type	Constraint(If any)
1	genre_id	INT	Primary key
2	name	VARCHAR	Not null
This table serves as a repository for book genres in an online bookstore database. It includes a unique identifier (“genre_id”) and a VARCHAR field (“name”) to store the genre names. It supports efficient genre identification, ensures data integrity through the primary key constraint, and is integral for organizing and categorizing books based on their genres.			

Book

S/N	Attribute	Data Type	Constraint(If any)
1	book_id	INT	Primary key
2	title	VARCHAR	Not null
3	publisher_id	INT	Not null, Foreign key
4	isbn	VARCHAR	Not null
5	number_of_pages	INT	Not null, number_of_pages > 0
6	language	VARCHAR	Not null
7	format	VARCHAR	Not null, format = 'Paperback' or format = 'Hardcover'
8	stock_quantity	INT	Not null, stock_quantity > 0
9	price	INT	Not null, price > 0
10	discount	DECIMAL	discount >= 0 and discount <= 100
11	ebook_available	BOOLEAN	Not null
12	publication_date	DATE	Not null
This table is a core component of the online bookstore database, storing detailed information about each book, including title, ISBN, language, format, pricing, and publication date. Its primary key (“book_id”) ensures unique identification, while foreign key (“publisher_id”) establishes a link to the “Publisher” table. Check constraints maintain data integrity, supporting inventory management (“stock_quantity”) and specifying electronic availability (“ebook_available”). It serves as a comprehensive repository for efficient book organization and retrieval.			

User

S/N	Attribute	Data Type	Constraint(If any)
1	user_id	INT	Primary key
2	name	VARCHAR	Not null
3	email	VARCHAR	Not null
4	username	VARCHAR	Not null
5	password	VARCHAR	Not null
6	address	VARCHAR	Not null
7	phone_number	VARCHAR	Not null
This table serves as a central repository for user-related information, supporting authentication, communication, and overall user management within the online bookstore system.			

Admin

S/N	Attribute	Data Type	Constraint(If any)
1	admin_id	INT	Primary key
2	name	VARCHAR	Not null
3	username	VARCHAR	Not null
4	password	VARCHAR	Not null
This table stores essential information about administrators. With fields for admin ID, name, username, and password, the table is structured with a primary key constraint to ensure unique identification of each administrator. This table supports secure access and management of administrative roles within the system.			

Orders

S/N	Attribute	Data Type	Constraint(If any)
1	order_id	VARCHAR	Primary key
2	user_id	INT	Not null, Foreign key
3	order_date	DATE	Not null
4	shipped_date	DATE	
5	address	VARCHAR	
6	phone_number	VARCHAR	
7	payment_status	VARCHAR	Not null, payment_status = 'Paid' or payment_status = 'Pending'
8	payment_method	VARCHAR	Not null, payment_method = 'Online' or payment_method = 'Cash'
9	total_price	INT	Not null, total_price > 0
This table in the online bookstore database tracks order details with a unique order ID, user association, and relevant dates. It enforces data integrity through constraints, ensuring valid payment status, methods, and positive total prices. This table is essential for efficient order management and financial tracking within the system.			

AuthorBook

S/N	Attribute	Data Type	Constraint(If any)
1	author_id	INT	Primary key, Foreign key
2	book_id	INT	Primary key, Foreign key
This table establishes author-book associations with a composite primary key, ensuring uniqueness and maintaining data integrity through foreign key constraints. It facilitates the representation of diverse authorship within the online bookstore database.			

GenreBook

S/N	Attribute	Data Type	Constraint(If any)
1	book_id	INT	Primary key, Foreign key
2	genre_id	INT	Primary key, Foreign key
This table enables the association between books and genres in the online bookstore database. With a composite primary key and foreign key constraints, it ensures data integrity by linking to the "Book" and "Genre" tables, facilitating organized categorization of books by genre.			

OrderBook

S/N	Attribute	Data Type	Constraint(If any)
1	order_id	VARCHAR	Primary key, Foreign key
2	book_id	INT	Primary key, Foreign key
3	quantity	INT	Not null
This table manages the relationship between orders and books in the online bookstore database. Featuring a composite primary key, it ensures uniqueness and maintains data integrity through foreign key constraints, connecting to the "Orders" and "Book" tables. This table enables the systematic tracking of ordered books within the system.			

Example Data:

Author

author_id	name	birthdate	country	email
1	Humayun Ahmed	1948-11-13	Bangladesh	humayun.ahmed@example.com
2	Taslima Nasrin	1962-08-25	Bangladesh	taslima.nasrin@example.com

Publisher

publisher_id	name	country	established_date
1	Anannya	Bangladesh	1970-01-01
2	Boi Bichitra	Bangladesh	1982-09-08

Genre

genre_id	name
1	Fiction
2	Non-Fiction

Book

bid	title	pid	isbn	page	language	format	quantity	price	discount	eb	pdate
1	Deyal	1	1234	350	Bengali	Hard cover	120	110	0.05	true	2022-03-01
12	Kuhok	2	1234	400	Bengali	Paper back	20	100	0.10	false	2022-02-2

User

user_id	name	email	username	password	address	phn_no
1	Md. Rahman	rahman@email.com	md_rahma	password123	Dhaka, Bangladesh	0123456789
2	Fatima Akter	akter@email.com	fatima_akter	password456	Chittagong	0123456789

Orders

order_id	user_id	order_date	shipped_date	payment_status	payment_method	total_price
0001000001	1	2023-01-01	2023-01-04	Paid	Online	5000
0002000001	2	2023-01-04	2023-01-30	Pending	Cash	2000

AuthorBook

author_id	book_id
1	1
1	2

GenreBook

genre_id	book_id
1	1
1	2

OrderBook

order_id	book_id	quantity
0001000001	1	2
0001000001	2	1

Admin

admin_id	name	username	password
1	Mahmudul Hasan	mahmudul	12345678
2	Fatima Akter	fatima_akter	password456

2.3 Implementation of Database Design

```
CREATE DATABASE BookStore;

CREATE TABLE Author (
    author_id INT,
    name VARCHAR(255) NOT NULL,
    birth_date DATE NOT NULL,
    country VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    CONSTRAINT auther_pk PRIMARY KEY (author_id)
);


CREATE TABLE Publisher (
    publisher_id INT,
    name VARCHAR(255) NOT NULL,
    country VARCHAR(255) NOT NULL,
    established_date DATE NOT NULL,
    CONSTRAINT publisher_pk PRIMARY KEY (publisher_id)
);

CREATE TABLE Genre (
    genre_id INT,
    name VARCHAR(255) NOT NULL,
    CONSTRAINT genre_pk PRIMARY KEY (genre_id)
);

CREATE TABLE Book (
    book_id INT,
    title VARCHAR(255) NOT NULL,
    publisher_id INT NOT NULL,
    isbn VARCHAR(20) NOT NULL,
```

```
    number_of_pages INT NOT NULL,
    language VARCHAR(50) NOT NULL,
    format VARCHAR(20) NOT NULL,
    stock_quantity INT NOT NULL,
    price INT NOT NULL,
    discount DECIMAL(5, 2),
    ebook_available BOOLEAN NOT NULL,
    publication_date DATE NOT NULL,
    CONSTRAINT book_pk PRIMARY KEY (book_id),
          CONSTRAINT   book_fk   FOREIGN   KEY   (publisher_id)   REFERENCES
Publisher(publisher_id),
    CONSTRAINT book_chk_page CHECK (number_of_pages > 0),
      CONSTRAINT  book_chk_format  CHECK  (format  =  'Paperback'  OR  format  =
'Hardcover'),
    CONSTRAINT book_chk_stock CHECK (stock_quantity > 0),
    CONSTRAINT book_chk_price CHECK (price > 0),
    CONSTRAINT book_chk_discount CHECK (discount >= 0 AND discount <= 100)
);


CREATE TABLE User (
    user_id INT,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    CONSTRAINT user_pk PRIMARY KEY (user_id)
);

CREATE TABLE Orders (
    order_id VARCHAR(10),
    user_id INT,
    order_date DATE,
    shipped_date DATE,
    address VARCHAR(255),
    phone_number VARCHAR(255),
    payment_status VARCHAR(50) NOT NULL,
    payment_method VARCHAR(50) NOT NULL,
    total_price INT NOT NULL,
    CONSTRAINT order_pk PRIMARY KEY (order_id),
    CONSTRAINT order_fk FOREIGN KEY (user_id) REFERENCES User(user_id),
```

```
        CONSTRAINT order_chk_status CHECK (payment_status = 'Paid' OR
payment_status = 'Pending'),
        CONSTRAINT order_chk_method CHECK (payment_method = 'Online' OR
payment_method = 'Cash'),
    CONSTRAINT order_chk_price CHECK (total_price > 0)
);

CREATE TABLE AuthorBook (
    author_id INT,
    book_id INT,
    CONSTRAINT authorbook_pk PRIMARY KEY (author_id, book_id),
      CONSTRAINT authorbook_author_fk FOREIGN KEY (author_id) REFERENCES
Author(author_id),
        CONSTRAINT authorbook_book_fk FOREIGN KEY (book_id) REFERENCES
Book(book_id)
);


DELIMITER //
CREATE FUNCTION getNewOrderNo(uid INT) RETURNS VARCHAR(10)
DETERMINISTIC
BEGIN
    DECLARE newOrderNo INT;

    SELECT COALESCE(MAX(CAST(SUBSTRING(order_id, 5) AS SIGNED)), 0) + 1
INTO newOrderNo
    FROM Orders
    WHERE user_id = uid;

    RETURN CONCAT(LPAD(CAST(uid AS CHAR), 4, '0'), LPAD(CAST(newOrderNo AS
CHAR), 6, '0'));
END //

DELIMITER //
CREATE TRIGGER trg_ord_bef_ins
BEFORE INSERT ON Orders
FOR EACH ROW
BEGIN
    SET NEW.order_id = getNewOrderNo(NEW.user_id);
END //


DELIMITER //
```

```
CREATE PROCEDURE addNewOrder(
    IN uid INT,
    IN odate DATE,
    IN sdate DATE,
    IN addrs VARCHAR(255),
    IN phn VARCHAR(255),
    IN pmt_status VARCHAR(50),
    IN pmt_method VARCHAR(50),
    IN price INT
)
BEGIN
    DECLARE newOrderNo VARCHAR(6);

    INSERT INTO Orders (user_id, order_date, shipped_date, address,
phone_number, payment_status, payment_method, total_price)
    VALUES
        (uid, odate, sdate, addrs, phn, pmt_status, pmt_method, price);
END //

CALL addNewOrder(1, '2023-06-01', '2023-06-05', 'Mirpur-1', '12345678901',
'Paid', 'Online', 5000.00);
```

Functional Dependencies, Normal Forms & Constraints

In crafting our Online Bookstore database, we've adhered diligently to the principles of normalization and functional dependencies. This ensures that our data remains free from redundancy, much like organizing books systematically on shelves without unnecessary repetition. By imposing constraints, we've established a set of rules akin to maintaining order in a library—no misplaced books, no incorrect information. This commitment to best practices guarantees a database that operates seamlessly, providing users with reliable and error-free access to a vast literary collection. Below is a detailed description of every table about their functional dependencies, normal forms & constraints.

3.1 Author

Functional Dependencies:

author_id -> name, birth_date, country, email

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here author_id is the primary key that ensures that each author has a unique identity.

3.2 Publisher

Functional Dependencies:
publisher_id -> name, country, established_date

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here publisher_id is the primary key that ensures that each publisher has a unique identity.

3.3 Genre

Functional Dependencies:
genre_id -> name

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here genre_id is the primary key that ensures that each genre has a unique identity.

3.4 Book

Functional Dependencies:
book_id -> title, publisher_id, isbn, number_of_pages, language, format, stock_quantity, price, discount, ebook_available, publication_date

Highest Normal Form:

This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
The primary key constraint on book_id ensures a unique identifier for each book. Publisher_id acts as a foreign key that references the publisher table's primary key. There is also some check constraints such as number_of_pages, stock_quantity, price have to be positive, format of the book should of two types ("Paperback", "Hardcover"), and discount should be a percentage between 0 to 100. Also before placing any orders, we will have to check internally that the stock_quantity is more than or equal to the ordered quantity. Before ordering a ebook version, we must ensure that ebook_available has value "true". When we add an existing value, the stock_quantity should be updated properly.

3.5 User

Functional Dependencies:
user_id -> name, email, username, password, address, phone_number

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here user_id is the primary key that ensures that each user has a unique identity. While letting a user to log in, his credentials i.e. username and password must be matched with stored credentials.

3.6 Orders

Functional Dependencies:
order_id -> user_id, order_date, shipped_date, address, phone_number, payment_status, payment_method, total_price

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here order_id is the primary key that ensures that each order has a unique identity. Foreign key constraint user_id references the User table's primary key to indicate

which user has placed the ordered. Payment status must of of two types (“Paid”, “Pending”) and payment_method also have to be of two types (“Online”, “Cash”). Total price have to be positive and must be calculated from the prices and discounts of the books that has been ordered. Shipped_date must be a date after the order_date. Orders table’s order_id is calculated using stored procedure based on user_id and his order counts.

3.7 AuthorBook

Functional Dependencies:
author_id, book_id -> author_id, book_id

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here author_id and book_id together acts as the primary key. The author_id and book_id both are foreign keys that references the primary key of the table Author and Book respectively.

3.8 GenreBook

Functional Dependencies:
genre_id, book_id -> genre_id, book_id

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here genre_id and book_id together acts as the primary key. The genre_id and book_id both are foreign keys that references the primary key of the table Genre and Book respectively.

3.9 OrderBook

Functional Dependencies:
order_id, book_id -> order_id, book_id, quantity

Highest Normal Form:

This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here order_id and book_id together acts as the primary key. The order_id and book_id both are foreign keys that references the primary key of the table Order and Book respectively.

3.10 Admin

Functional Dependencies:
admin_id -> name, username, password

Highest Normal Form:
This table is in **Boyce-Codd Normal Form (BCNF)** as every functional dependency of the table is a dependency on a candidate key.

Constraints:
Here admin_id is the primary key that ensures that each admin has a unique identity. While letting a admin to log in, his credentials i.e. username and password must be matched with stored credentials.

Expected Queries & Implementations

4.1 Search books based on writer's name

MySQL Statement:

```
SELECT *
FROM Book
JOIN AuthorBook ON Book.book_id = AuthorBook.book_id
JOIN Author ON AuthorBook.author_id = Author.author_id
WHERE Author.name = 'AuthorName';
```

Example Output:
When AuthorName = "Humayun Ahmed" (Only showing book_id, title and 2 rows)

book_id	title
1	Deyal

11	Onishchito

4.2 Search books based on genre

MySQL Statement:

```
SELECT *
FROM Book
JOIN GenreBook ON Book.book_id = GenreBook.book_id
JOIN Genre ON GenreBook.genre_id = Genre.genre_id
WHERE Genre.name = 'GenreName';
```

Example Output:
When GenreName = “Fiction” (Only showing book_id, title and 2 rows)

book_id	title
1	Deyal
2	Amar Ache Jol

4.3 Search books based on publication’s name

MySQL Statement:

```
SELECT DISTINCT Book.*
FROM Book
JOIN Publisher ON Book.publisher_id = Publisher.publisher_id
WHERE Publisher.name = 'PublicationName';
```

Example Output:
When PublicationName = “Anannya” (Only showing book_id, title and 2 rows)

book_id	title
1	Deyal
11	Onishchito

4.4 Search latest books

MySQL Statement:

```
SELECT *
FROM Book
ORDER BY publication_date DESC;
```

Example Output:
(Only showing book_id, title and 2 rows)

book_id	title
17	Himu Mama
13	Rupali Dip

4.5 Search books that have special offer

MySQL Statement:

```
SELECT *
FROM Book
WHERE discount > special_offer_percentage;
```

Example Output:
When special_offer_percentage = 0.1 (Only showing book_id, title and 2 rows)

book_id	title
11	Onishchito
13	Rupali Dip

4.6 Search books that have ebook available

MySQL Statement:

```
SELECT *
FROM Book
WHERE ebook_available = true;
```

Example Output:
(Only showing book_id, title and 2 rows)

book_id	title

1	Deyal
2	Amar Ache Jol

4.6 Search books on the basis of price

MySQL Statement:

```
SELECT *
FROM Book
ORDER BY price ASC;
```

Example Output:
(Only showing book_id, title and 2 rows)

book_id	title
2	Amar Ache Jol
3	Anil Bagchir Ekdin

4.7 Search books that have been ordered multiple times

MySQL Statement:

```
SELECT b.*
FROM Book b
JOIN OrderBook ob ON b.book_id = ob.book_id
GROUP BY b.book_id
HAVING COUNT(ob.order_id) > Number;
```

Example Output:
When Number = 3 (Only showing book_id, title and 2 rows)

book_id	title
1	Deyal
2	Amar Ache Jol

4.8 Search number of copies available of a book

MySQL Statement:

```
SELECT stock_quantity
FROM Book
WHERE book_id = searchedBookId;
```

Example Output:
When searchedBookId = 1

stock_quantity
120

4.9 Yearly bestseller books

MySQL Statement:

```
SELECT ob.book_id, b.title
FROM OrderBook ob
JOIN Book b ON ob.book_id = b.book_id
JOIN Orders od ON od.order_id = ob.order_id
WHERE YEAR(od.order_date) = YEAR(NOW())
GROUP BY ob.book_id
ORDER BY COUNT(*) DESC;
```

Example Output:
(Only showing 2 rows)

book_id	title
1	Deyal
2	Amar Ache Jol

4.10 Yearly bestseller authors

MySQL Statement:

```
SELECT a.author_id, a.name
FROM OrderBook ob
JOIN Book b ON ob.book_id = b.book_id
```

```
JOIN Orders od ON od.order_id = ob.order_id
JOIN AuthorBook ab ON b.book_id = ab.book_id
JOIN Author a ON ab.author_id = a.author_id
WHERE YEAR(od.order_date) = YEAR(NOW())
GROUP BY a.author_id
ORDER BY COUNT(*) DESC;
```

Example Output:
(Only showing 2 rows)

author_id	name
1	Humayun Ahmed
3	Muhammed Zafar Iqbal

4.11 Suggest books that have been bought together

MySQL Statement:

```
SELECT b2.book_id, b2.title
FROM OrderBook ob1
JOIN OrderBook ob2 ON ob1.order_id = ob2.order_id AND ob1.book_id <>
ob2.book_id
JOIN Book b1 ON ob1.book_id = b1.book_id
JOIN Book b2 ON ob2.book_id = b2.book_id
WHERE b1.book_id = searchedBookId
GROUP BY b2.book_id, b2.title;
```

Example Output:
When searchedBookId = 1 (Only showing 2 rows)

book_id	title
2	Amar Ache Jol
3	Anil Bagchir Ekdin

4.12 Top 3 customers with who spent the most

MySQL Statement:

```
SELECT u.name, SUM(o.total_price) as total_spent
```

```
FROM User u
JOIN Orders o ON u.user_id = o.user_id
GROUP BY u.user_id
ORDER BY total_spent DESC
LIMIT 3;
```

Example Output:

name	total_spent
Md Rahman	27200
Tasnim Hossain	134000
Abdul Karim	115000

4.13 Most popular genres

MySQL Statement:

```
SELECT
    g.genre_id,
    g.name AS genre_name
FROM
    Genre g
JOIN GenreBook bg ON
    g.genre_id = bg.genre_id
GROUP BY
    g.genre_id, genre_name
ORDER BY
    COUNT(bg.book_id) DESC;
```

Example Output:

genre_id	genre_name
1	Fiction
2	Non Fiction
3	Mystery

4.14 Show total sales of each book

MySQL Statement:

```
SELECT
    b.book_id,
    b.title,
    SUM(ob.quantity) AS total_sales
FROM
    Book b
JOIN OrderBook ob ON
    b.book_id = ob.book_id
GROUP BY
    b.book_id, b.title;
```

Example Output:
(Only showing 2 rows)

book_id	title	total_sales
1	Deyal	8
2	Amar Ache Jol	8

4.15 Percentage of orders that paid through online services

MySQL Statement:

```
SELECT
    (COUNT(CASE WHEN o.payment_method = 'Online' THEN 1 END) * 100.0 /
COUNT(*)) AS percentage_online_payments
FROM
    Orders o;
```

Example Output:

percentage_online_payments
63.63636

4.16 Show order history

MySQL Statement:

```
SELECT
    o.order_id,
    b.title,
    ob.quantity
FROM
    Orders o
JOIN
    OrderBook ob ON o.order_id = ob.order_id
JOIN
    Book b ON ob.book_id = b.book_id
WHERE
    o.user_id = userId;
```

Example Output:
When userId = 1 (Only showing 2 rows)

order_id	title	quantity
0001000001	Deyal	2
0001000001	Amar Ache Jol	1

4.17 Books that have not been ordered yet

MySQL Statement:

```
SELECT
    b.book_id,
    b.title
FROM
    Book b
LEFT JOIN
    OrderBook ob ON b.book_id = ob.book_id
WHERE
    ob.book_id IS NULL;
```

Example Output:

book_id	title
21	Halum

4.18 User have not placed any order yet

MySQL Statement:

```
SELECT
    u.user_id,
    u.name
FROM
    User u
LEFT JOIN
    Orders o ON u.user_id = o.user_id
WHERE
    o.user_id IS NULL;
```

Example Output:

user_id	name
11	Md. Rahman

4.19 Show the revenue generated by each author

MySQL Statement:

```
SELECT
    a.author_id,
    a.name AS author_name,
    SUM((b.price * (1 - COALESCE(b.discount, 0) / 100)) * ob.quantity) AS
total_revenue
FROM
    Author a
JOIN
    AuthorBook ab ON a.author_id = ab.author_id
JOIN
    Book b ON ab.book_id = b.book_id
JOIN
    OrderBook ob ON b.book_id = ob.book_id
JOIN
    Orders o ON ob.order_id = o.order_id
WHERE
    o.payment_status = 'Paid'
GROUP BY
    a.author_id, a.name
ORDER BY
```

```
    total_revenue DESC;
```

Example Output:
(Only showing 2 rows)

author_id	author_name	total_revenue
8	Shirshendu Mukhopadhyay	15383.601000
1	Humayun Ahmed	14673.558000

4.20 Show orders that are yet to be shipped

MySQL Statement:

```
SELECT
    order_id,
    order_date
FROM
    Orders
WHERE
    shipped_date IS NULL;
```

Example Output:
(Only showing 2 rows)

order_id	order_date
0001000003	2023-01-21
0002000003	2023-01-22

Conclusion

The development of the Online Bookstore database has laid a solid foundation for a comprehensive and efficient platform. The carefully designed schema, adhering to normalization principles, functional dependencies, and constraints, ensures data integrity and minimizes redundancy. The entities, from Authors and Publishers to Books and Users, seamlessly interact to create a robust ecosystem for managing, exploring, and ordering books.

Moving forward, the integration of a user interface (UI) will be a pivotal step to enhance the user experience. The UI will provide a visually appealing and user-friendly interface, allowing customers to easily navigate through the extensive collection of books, place orders, and interact with the bookstore seamlessly. The addition of features such as search functionalities, personalized user profiles, and an intuitive ordering system will further elevate the overall usability of the Online Bookstore.

Future plans also encompass the implementation of data analytics to derive meaningful insights from the database. This can include tracking popular genres, identifying best-selling authors, and analyzing customer purchasing patterns. Such insights can inform strategic decisions, improve inventory management, and enhance the bookstore's offerings.

Overall, the Online Bookstore project stands poised for further evolution, with a strategic focus on user interface development and the integration of advanced analytics to provide an enriched and tailored experience for book enthusiasts.