Reichman University

Relational Databases and SQL 2025

Assignment #3

THIS ASSIGNMENT CAN BE SUBMITTED INDIVIDUALLY OR IN PAIRS.

Assignment Overview

In this assignment, you will design a simple relational database based on a provided use case. You will create tables with appropriate data types and constraints, establish relationships between the tables, and populate them with sample data.

Use Case: Online Bookstore

You are tasked with creating a database for an online bookstore. The bookstore needs to keep track of its books, authors, customers, and orders. Below are the detailed requirements for each entity and the associated business rules that must be enforced through constraints.

1. **Books**
    - **BookID**: A unique identifier for each book.
    - **Title**: The title of the book.
    - **Genre**: The genre of the book (e.g., Fiction, Non-Fiction, Science Fiction, Biography).
    - **Price**: The selling price of the book.
        - Price must be a positive value.
    - **StockQuantity**: The number of copies available in the store.
        - Stock Quantity cannot be negative.
    - **PublicationDate**: The date the book was published.

2. **Authors**
    - **AuthorID**: A unique identifier for each author.
    - **FirstName**: Author's first name.
    - **LastName**: Author's last name.
    - **Email**: Contact email for the author.
        - Email must be unique.

3. **Customers**
    - **CustomerID**: A unique identifier for each customer.
    - **FirstName**: Customer's first name.
    - **LastName**: Customer's last name.
    - **Email**: Contact email for the customer.
        - Email must be unique.
    - **PhoneNumber**: Customer's phone number.
        - Phone number must be unique.
        - Phone number should have exactly 10 digits.
    - **Address**: Customer's physical address.

4. **Orders**
    - **OrderID**: A unique identifier for each order.
    - **CustomerID**: References the customer who placed the order.
        - Must reference an existing Customer ID.
    - **OrderDate**: The date when the order was placed.
    - **ShippingDate:** The date when the order was shipped.
        - Must be on or after the OrderDate (an order cannot be shipped before it is placed).
        - Can remain empty for orders that have not been shipped yet.
    - **OrderStatus**: The current status of the order.
        - Possible values: Pending, Completed, Cancelled.
    - **PaymentMethod**: The method used for payment.
        - Possible values: Credit Card, PayPal, Cash on Delivery.

5. **Order_Items**

   This table is used to record the details of books included in each order.
   It acts as a bridge between the Orders and Books tables.

   - **OrderItemID**: A unique identifier for each order item.
   - **OrderID**: References the order.
     - Must reference an existing Order ID.
   - **BookID**: References the book being ordered.
     - Must reference an existing Book ID.
   - **Quantity**: Number of copies of the book in the order.
     - Quantity must be at least 1.
   - **UnitPrice**: Price per copy at the time of the order.
     - Unit Price must be a positive value.

<u>Constraint for Order_Items:</u> Ensure that the same book (Book ID) cannot appear multiple times within a single order (Order ID). (If a customer orders multiple copies of the same book, the "Quantity" column should reflect the total number of copies instead of creating duplicate rows).

6. **Book_Authors**
   - **BookAuthorID:** A synthetic primary key for the table.
   - **BookID**: References the book.
     - Must reference an existing Book ID.
   - **AuthorID**: References the author.
     - Must reference an existing Author ID.
   - Combination of Book ID and Author ID must be unique to prevent duplicate author assignments to the same book.

Note: The Book_Authors table helps us track which authors wrote which books. This table is needed because a book can have multiple authors, and an author can write multiple books.

Tasks

1. **Design the Database Schema**
    a. **Create Tables**
        - Define tables for Books, Authors, Customers, Orders, Order_Items, and Book_Authors based on the use case.
        - Choose appropriate data types for each column.

    b. **Define Primary Keys**
        - Assign a primary key to each table.

    c. **Establish Relationships and Foreign Keys**
        - Set up foreign keys to represent relationships between tables.
        - Ensure referential integrity using FOREIGN KEY constraints.

    d. **Apply Constraints**
        - Implement NOT NULL constraints where applicable.
        - Apply UNIQUE constraints for fields like Email.
        - Implement additional constraints based on the detailed business rules provided above (e.g., CHECK constraints for positive values).

2. **Insert Sample Data**
    a. Populate each table with at least **5** sample records.
    b. Ensure that foreign key relationships are respected (e.g., CustomerID in Orders must exist in Customers).
    c. Ensure that all constraints are satisfied with the sample data (e.g., no negative Stock Quantities).

3. **SQL Scripts Submission**
    a. Submit a single SQL script file that contain only the following types of commands:
        - CREATE TABLE statements to define each table with its constraints.
        - INSERT INTO statements to populate each table with sample data.
    b. **Do Not Include:**
        - Any SELECT, UPDATE, DELETE, or other types of queries.
        - Testing queries or additional operations outside of table creation and data insertion.

Submission Guidelines

- Ensure your SQL script runs without errors on **MySQL Workbench**.
- Comment your SQL script for clarity where necessary. One of the evaluation criteria here is your code quality, so make sure your SQL script is well-organized, readable, and properly commented.
- Standardize the database name as it's described below.
- Submit the SQL script file via the course's submission portal by the due date.

If submitting in pairs, both students must adhere to the following guidelines:

- Each student must submit the assignment **SQL File** through their respective Moodle accounts.
- The file names must be **identical** in both submissions and must include both student ID numbers (e.g., `123456789_987654321.pdf` and `123456789_987654321.sql`).

Standardize the Database Name

To ensure consistency across all submissions, include the following lines at the beginning of your SQL script. This will reset the database each time the script is run and help you to debug your script:

```
DROP DATABASE IF EXISTS OnlineBookstoreDB;

CREATE DATABASE OnlineBookstoreDB;

USE OnlineBookstoreDB;
```

Tips

1. **Testing Constraints:** After creating the tables and inserting data, try inserting records that violate constraints to ensure that they are enforced correctly. This is a good way to understand the importance of each constraint.

2. **Understanding Data Types:** Pay attention to choosing appropriate data types. For example, using DECIMAL for monetary values helps maintain precision.

3. **Maintaining Referential Integrity:** Always ensure that foreign keys reference existing primary keys to prevent orphaned records.

Good luck, and feel free to reach out if you have any questions! 🙂