**Note:** In latest mysql version, constraints on date is not possible, You can ignore if any questions on date

**Exercise 1: Creating and Populating the Authors Table**

**Exercise:**

1. Create a table named authors with the following structure:
   * author\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Unique.
   * date\_of\_birth: Date.
2. Insert the following records:
   * ('J.K.', 'Rowling', 'jk.rowling@example.com', '1965-07-31')
   * ('George', 'Orwell', 'george.orwell@example.com', '1903-06-25')
   * ('Jane', 'Austen', 'jane.austen@example.com', '1775-12-16')
3. Try to insert a duplicate email ('Herman', 'Melville', 'jane.austen@example.com', '1819-08-01').

**Exercise 2: Ensuring Positive Prices in the Books Table**

**Exercise:**

1. Create a table named books with the following structure:
   * book\_id: Integer, Auto Increment, Primary Key.
   * title: Variable character, Not Null.
   * author\_id: Integer, Foreign Key referencing authors.author\_id.
   * isbn: Variable character, Unique, Not Null.
   * published\_date: Date.
   * price: Decimal, greater than 0 (Use CHECK constraint).
   * available\_copies: Integer, greater than or equal to 0 (Use CHECK constraint).
2. Insert the following records:
   * ('Harry Potter and the Goblet of Fire', 1, '9780439139601', '2000-07-08', 22.99, 5)
   * ('1984', 2, '9780451524935', '1949-06-08', 15.50, 10)
3. Try to insert a book with a negative price ('Animal Farm', 2, '9780451526342', '1945-08-17', -10.00, 5).

**Exercise 3: Managing Library Members with Constraints**

**Exercise:**

1. Create a table named members with the following structure:
   * member\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * phone: Variable character, optional.
   * join\_date: Date, default to current date.
2. Insert the following records:
   * ('Alice', 'Johnson', 'alice.johnson@example.com', '1234567890')
   * ('Bob', 'Smith', 'bob.smith@example.com', '0987654321')
3. Try to insert a member with a NULL email ('Charlie', 'Brown', NULL, '1122334455').

**Exercise 4: Borrowing Records with Date Constraints**

**Exercise:**

1. Create a table named borrow\_records with the following structure:
   * borrow\_id: Integer, Auto Increment, Primary Key.
   * member\_id: Integer, Foreign Key referencing members.member\_id.
   * book\_id: Integer, Foreign Key referencing books.book\_id.
   * borrow\_date: Date, Not Null.
   * return\_date: Date.
   * due\_date: Date, Not Null.
   * Check constraint to ensure that return\_date is either NULL or greater than or equal to borrow\_date.
2. Insert the following records:
   * member\_id = 1, book\_id = 1, borrow\_date = '2024-09-01', return\_date = '2024-09-15', due\_date = '2024-09-30'
   * member\_id = 2, book\_id = 2, borrow\_date = '2024-09-05', return\_date = NULL, due\_date = '2024-09-25'
3. Try to insert a record with return\_date before borrow\_date member\_id = 1, book\_id = 1, borrow\_date = '2024-09-10', return\_date = '2024-09-05', due\_date = '2024-09-20'.

**Exercise 5: Complex Insert and Constraint Check**

**Exercise:**

1. Insert a new author ('Dan', 'Brown', 'dan.brown@example.com', '1964-06-22').
2. Insert a new book ('The Da Vinci Code', 5, '9780307474278', '2003-03-18', 12.99, 7).
3. Insert a member ('Emma', 'Watson', 'emma.watson@example.com', '5551234567').
4. Create a borrowing record for the newly created member and book, with borrow\_date being today and due\_date being 15 days later.

**Exercise 6: Managing a Movie Database**

**Exercise:**

1. Create a table named directors with the following structure:
   * director\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * birth\_date: Date, Not Null.
   * country: Variable character, Not Null, Default to 'Unknown'.
2. Create a table named movies with the following structure:
   * movie\_id: Integer, Auto Increment, Primary Key.
   * title: Variable character, Not Null.
   * release\_year: Year, Not Null, Check constraint to ensure year is greater than or equal to 1900.
   * genre: Variable character, Not Null, Default to 'Drama'.
   * director\_id: Integer, Foreign Key referencing directors.director\_id.
   * rating: Decimal(3,1), Not Null, Check constraint to ensure value is between 0 and 10.
   * Unique constraint on (title, release\_year).
3. Insert the following directors:
   * ('Steven', 'Spielberg', '1946-12-18', 'USA')
   * ('Christopher', 'Nolan', '1970-07-30', 'UK')
   * ('Quentin', 'Tarantino', '1963-03-27', 'USA')
   * ('Unknown', 'Director', '1900-01-01', NULL) (will use default value for country)
4. Insert the following movies:
   * ('Inception', 2010, 'Sci-Fi', 2, 8.8)
   * ('Jurassic Park', 1993, 'Adventure', 1, 8.1)
   * ('Pulp Fiction', 1994, 'Crime', 3, 8.9)
5. Try to insert a movie with a year less than 1900 and a rating greater than 10.

**Exercise 7: Employee-Department Relationship with Salary Constraints**

**Exercise:**

1. Create a table named departments with the following structure:
   * department\_id: Integer, Auto Increment, Primary Key.
   * department\_name: Variable character, Unique, Not Null.
2. Create a table named employees with the following structure:
   * employee\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * department\_id: Integer, Foreign Key referencing departments.department\_id.
   * hire\_date: Date, Not Null, Default to the current date.
   * salary: Decimal(10,2), Not Null, Check constraint to ensure salary is greater than 2000.
   * Unique constraint on (first\_name, last\_name).
3. Insert the following departments:
   * ('HR')
   * ('Finance')
   * ('IT')
   * ('Sales')
4. Insert the following employees:
   * ('Alice', 'Johnson', 1, '2024-01-10', 5000.00)
   * ('Bob', 'Smith', 2, '2024-03-15', 4500.00)
   * ('Charlie', 'Brown', 3, NULL, 3000.00) (uses default hire\_date)
   * ('David', 'Wilson', 3, NULL, 1800.00) (This should fail due to salary constraint)

**Exercise 8: Inventory Management System**

**Exercise:**

1. Create a table named suppliers with the following structure:
   * supplier\_id: Integer, Auto Increment, Primary Key.
   * supplier\_name: Variable character, Unique, Not Null.
   * contact\_name: Variable character, Not Null.
   * phone: Variable character, Not Null.
   * email: Variable character, Not Null, Unique.
2. Create a table named products with the following structure:
   * product\_id: Integer, Auto Increment, Primary Key.
   * product\_name: Variable character, Unique, Not Null.
   * supplier\_id: Integer, Foreign Key referencing suppliers.supplier\_id.
   * unit\_price: Decimal(10, 2), Not Null, Check constraint to ensure price is greater than 0.
   * units\_in\_stock: Integer, Not Null, Check constraint to ensure value is greater than or equal to 0.
3. Insert the following suppliers:
   * ('Supplier A', 'John Doe', '123-456-7890', 'contact@supplierA.com')
   * ('Supplier B', 'Jane Doe', '234-567-8901', 'contact@supplierB.com')
4. Insert the following products:
   * ('Product 1', 1, 20.00, 50)
   * ('Product 2', 2, 15.50, 100)
   * ('Product 3', 1, -5.00, 30) (This should fail due to negative price)
   * ('Product 4', 2, 30.00, -10) (This should fail due to negative stock)

**Exercise 9: E-commerce Order Management System**

**Exercise:**

1. Create a table named customers with the following structure:
   * customer\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Not Null, Unique.
   * phone: Variable character, Unique, Not Null.
   * registration\_date: Date, Default to current date.
2. Create a table named products with the following structure:
   * product\_id: Integer, Auto Increment, Primary Key.
   * product\_name: Variable character, Unique, Not Null.
   * price: Decimal(10, 2), Not Null, Check constraint to ensure price is greater than 0.
   * stock\_quantity: Integer, Not Null, Check constraint to ensure quantity is greater than or equal to 0.
   * category: Variable character, Not Null, Default to 'General'.
3. Create a table named orders with the following structure:
   * order\_id: Integer, Auto Increment, Primary Key.
   * customer\_id: Integer, Foreign Key referencing customers.customer\_id.
   * order\_date: Date, Not Null, Default to current date.
   * total\_amount: Decimal(10, 2), Not Null, Default to 0.
4. Create a table named order\_items with the following structure:
   * order\_item\_id: Integer, Auto Increment, Primary Key.
   * order\_id: Integer, Foreign Key referencing orders.order\_id.
   * product\_id: Integer, Foreign Key referencing products.product\_id.
   * quantity: Integer, Not Null, Check constraint to ensure quantity is greater than 0.
   * item\_price: Decimal(10, 2), Not Null, Check constraint to ensure price is greater than 0.
   * Unique constraint on (order\_id, product\_id).
5. Insert the following customers:
   * ('John', 'Doe', 'john.doe@example.com', '1234567890')
   * ('Jane', 'Smith', 'jane.smith@example.com', '0987654321')
6. Insert the following products:
   * ('Laptop', 999.99, 10, 'Electronics')
   * ('Headphones', 49.99, 50, 'Accessories')
   * ('Coffee Mug', 9.99, 100, 'Kitchen')
7. Insert an order for John Doe with the following items:
   * 2 Laptops at $999.99 each.
   * 3 Coffee Mugs at $9.99 each.
8. Update the orders.total\_amount field to reflect the correct total for the above order.
9. Try to insert an order item with a quantity of 0 and a price of -5.

**Exercise 10: University Enrollment System**

**Exercise:**

1. Create a table named students with the following structure:
   * student\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Not Null, Unique.
   * birthdate: Date, Not Null.
   * enrollment\_date: Date, Default to current date.
2. Create a table named courses with the following structure:
   * course\_id: Integer, Auto Increment, Primary Key.
   * course\_name: Variable character, Unique, Not Null.
   * credits: Integer, Not Null, Check constraint to ensure credits are between 1 and 5.
3. Create a table named enrollments with the following structure:
   * enrollment\_id: Integer, Auto Increment, Primary Key.
   * student\_id: Integer, Foreign Key referencing students.student\_id.
   * course\_id: Integer, Foreign Key referencing courses.course\_id.
   * enrollment\_date: Date, Not Null, Default to current date.
   * Unique constraint on (student\_id, course\_id).
4. Insert the following students:
   * ('Alice', 'Williams', 'alice.williams@example.com', '2000-05-15')
   * ('Bob', 'Johnson', 'bob.johnson@example.com', '1999-10-20')
   * ('Charlie', 'Davis', 'charlie.davis@example.com', '1998-02-28')
5. Insert the following courses:
   * ('Mathematics', 3)
   * ('Physics', 4)
   * ('Chemistry', 5)
6. Enroll Alice Williams and Bob Johnson in the Mathematics course.
7. Enroll Alice Williams in the Physics course twice. (This should fail due to unique constraint.)

**Exercise 11: Hospital Management System**

**Exercise:**

1. Create a table named doctors with the following structure:
   * doctor\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * specialization: Variable character, Not Null.
   * phone: Variable character, Unique, Not Null.
   * email: Variable character, Unique, Not Null.
2. Create a table named patients with the following structure:
   * patient\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * dob: Date, Not Null.
   * gender: Enum ('Male', 'Female', 'Other'), Not Null.
   * phone: Variable character, Unique, Not Null.
   * email: Variable character, Unique, Not Null.
3. Create a table named appointments with the following structure:
   * appointment\_id: Integer, Auto Increment, Primary Key.
   * doctor\_id: Integer, Foreign Key referencing doctors(doctor\_id).
   * patient\_id: Integer, Foreign Key referencing patients(patient\_id).
   * appointment\_date: Date, Not Null.
   * appointment\_time: Time, Not Null.
   * status: Enum ('Scheduled', 'Completed', 'Canceled'), Not Null.
   * Unique constraint on (doctor\_id, appointment\_date, appointment\_time).
4. Create a table named prescriptions with the following structure:
   * prescription\_id: Integer, Auto Increment, Primary Key.
   * appointment\_id: Integer, Foreign Key referencing appointments(appointment\_id).
   * medicine\_name: Variable character, Not Null.
   * dosage: Variable character, Not Null.
   * duration: Integer, Not Null Check constraint to ensure duration is greater than 0.
   * Unique constraint on (appointment\_id, medicine\_name).
5. Insert the following doctors:
   * ('John', 'Doe', 'Cardiologist', '1234567890', 'john.doe@hospital.com')
   * ('Emily', 'Clark', 'Dermatologist', '0987654321', 'emily.clark@hospital.com')
6. Insert the following patients:
   * ('Alice', 'Johnson', '1990-04-15', 'Female', '2223334444', 'alice.johnson@gmail.com')
   * ('Bob', 'Smith', '1985-12-30', 'Male', '5556667777', 'bob.smith@yahoo.com')
7. Schedule the following appointments:
   * Alice Johnson with Dr. John Doe on 2024-10-01 at 10:00:00.
   * Bob Smith with Dr. Emily Clark on 2024-10-02 at 14:00:00.
8. Insert a prescription for Alice's appointment with the following details:
   * Medicine: Aspirin, Dosage: 1 tablet twice daily, Duration: 7 days.
9. Try to schedule another appointment for Dr. John Doe with Bob Smith on 2024-10-01 at 10:00:00 (This should fail due to unique constraint).
10. Try to insert a prescription with a duration of -5 (This should fail due to check constraint).

**Exercise 12: E-commerce Platform**

**Exercise:**

1. **Create a table named users:**
   * user\_id: Integer, Auto Increment, Primary Key.
   * username: Variable character, Unique, Not Null.
   * password: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * created\_at: Timestamp, Default to current timestamp.
2. **Create a table named products:**
   * product\_id: Integer, Auto Increment, Primary Key.
   * product\_name: Variable character, Unique, Not Null.
   * description: Text, Not Null.
   * price: Decimal(10, 2), Not Null, Check constraint to ensure price is greater than 0.
   * stock\_quantity: Integer, Not Null, Check constraint to ensure stock is greater than or equal to 0.
3. **Create a table named orders:**
   * order\_id: Integer, Auto Increment, Primary Key.
   * user\_id: Integer, Foreign Key referencing users(user\_id).
   * order\_date: Timestamp, Default to current timestamp.
   * status: Enum('Pending', 'Shipped', 'Delivered', 'Canceled'), Not Null.
4. **Create a table named order\_items:**
   * order\_item\_id: Integer, Auto Increment, Primary Key.
   * order\_id: Integer, Foreign Key referencing orders(order\_id).
   * product\_id: Integer, Foreign Key referencing products(product\_id).
   * quantity: Integer, Not Null, Check constraint to ensure quantity is greater than 0.
   * Unique constraint on (order\_id, product\_id).
5. **Insert the following users:**
   * ('alice123', 'password123', 'alice@example.com')
   * ('bob456', 'password456', 'bob@example.com')
6. **Insert the following products:**
   * ('Laptop', 'High-performance laptop', 999.99, 50)
   * ('Smartphone', 'Latest model smartphone', 699.99, 100)
7. **Create an order for Alice and add items to the order:**
   * Alice places an order with ID 1, which includes:
     + 1 Laptop
     + 2 Smartphones
8. **Try to add the same product to Alice's order again (This should fail due to the unique constraint).**
9. **Try to insert a product with a price of -10 (This should fail due to the check constraint).**
10. **Cancel Alice's order and change the status accordingly.**

**Exercise 13: Event Management System**

**Exercise:**

1. **Create a table named organizers:**
   * organizer\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * phone: Variable character, Unique, Not Null.
2. **Create a table named venues:**
   * venue\_id: Integer, Auto Increment, Primary Key.
   * venue\_name: Variable character, Not Null.
   * capacity: Integer, Not Null, Check constraint to ensure capacity is greater than 0.
   * location: Variable character, Not Null.
3. **Create a table named events:**
   * event\_id: Integer, Auto Increment, Primary Key.
   * event\_name: Variable character, Not Null.
   * organizer\_id: Integer, Foreign Key referencing organizers(organizer\_id).
   * venue\_id: Integer, Foreign Key referencing venues(venue\_id).
   * event\_date: Date, Not Null.
   * status: Enum('Scheduled', 'Completed', 'Canceled'), Not Null.
4. **Create a table named tickets:**
   * ticket\_id: Integer, Auto Increment, Primary Key.
   * event\_id: Integer, Foreign Key referencing events(event\_id).
   * ticket\_type: Variable character, Not Null.
   * price: Decimal(10, 2), Not Null, Check constraint to ensure price is greater than 0.
   * available\_quantity: Integer, Not Null, Check constraint to ensure quantity is greater than or equal to 0.
5. **Insert the following organizers:**
   * ('John Events', 'john.events@example.com', '1234567890')
   * ('Anna Concerts', 'anna.concerts@example.com', '0987654321')
6. **Insert the following venues:**
   * ('Stadium A', 5000, 'City Center')
   * ('Convention Hall B', 2000, 'Downtown')
7. **Create an event for John Events in Stadium A:**
   * ('Music Fest', 1, 1, '2024-11-15', 'Scheduled')
8. **Create tickets for the event:**
   * ('VIP', 199.99, 100)
   * ('Regular', 99.99, 200)
9. **Try to insert a ticket with a negative price (This should fail due to the check constraint).**
10. **Change the status of the event to Completed.**

**Exercise 14: Library Management System**

**Exercise:**

1. **Create a table named members:**
   * member\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * phone: Variable character, Unique, Not Null.
   * membership\_date: Date, Not Null.
2. **Create a table named books:**
   * book\_id: Integer, Auto Increment, Primary Key.
   * title: Variable character, Unique, Not Null.
   * author: Variable character, Not Null.
   * isbn: Variable character, Unique, Not Null.
   * copies\_available: Integer, Not Null, Check constraint to ensure greater than or equal to 0.
3. **Create a table named loans:**
   * loan\_id: Integer, Auto Increment, Primary Key.
   * member\_id: Integer, Foreign Key referencing members(member\_id).
   * book\_id: Integer, Foreign Key referencing books(book\_id).
   * loan\_date: Date, Not Null.
   * return\_date: Date, Nullable.
   * Check constraint to ensure return\_date is after loan\_date.
4. **Insert the following members:**
   * ('John Doe', 'john@example.com', '1234567890', '2024-01-15')
   * ('Jane Smith', 'jane@example.com', '0987654321', '2024-02-10')
5. **Insert the following books:**
   * ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 5)
   * ('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 3)
6. **Record a loan for John Doe:**
   * John borrows "The Great Gatsby" on 2024-09-01.
7. **Try to insert a loan for Jane Smith with a return date that is before the loan date (This should fail due to the check constraint).**
8. **Update the loan for John to set the return date to 2024-09-10.**
9. **Try to insert a book with a negative number of available copies (This should fail due to the check constraint).**

**Exercise 15: School Management System**

**Exercise:**

1. **Create a table named students:**
   * student\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Not Null.
   * dob: Date, Not Null.
   * email: Variable character, Unique, Not Null.
2. **Create a table named courses:**
   * course\_id: Integer, Auto Increment, Primary Key.
   * course\_name: Variable character, Unique, Not Null.
   * credits: Integer, Not Null, Check constraint to ensure greater than 0.
3. **Create a table named enrollments:**
   * enrollment\_id: Integer, Auto Increment, Primary Key.
   * student\_id: Integer, Foreign Key referencing students(student\_id).
   * course\_id: Integer, Foreign Key referencing courses(course\_id).
   * enrollment\_date: Date, Not Null.
4. **Insert the following students:**
   * ('Michael Johnson', '2000-05-15', 'michael.johnson@example.com')
   * ('Emily Davis', '1999-09-20', 'emily.davis@example.com')
5. **Insert the following courses:**
   * ('Mathematics', 3)
   * ('History', 4)
6. **Enroll Michael in Mathematics:**
   * Enrollment date is 2024-09-01.
7. **Try to enroll Emily in History again (This should fail due to the unique constraint).**
8. **Insert an invalid course with a negative number of credits (This should fail).**

**Exercise 16: Food Delivery System**

**Exercise:**

1. **Create a table named restaurants:**
   * restaurant\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Unique, Not Null.
   * address: Variable character, Not Null.
   * rating: Decimal(2, 1), Check constraint to ensure rating is between 0 and 5.
2. **Create a table named menu\_items:**
   * item\_id: Integer, Auto Increment, Primary Key.
   * restaurant\_id: Integer, Foreign Key referencing restaurants(restaurant\_id).
   * item\_name: Variable character, Not Null.
   * price: Decimal(10, 2), Not Null, Check constraint to ensure greater than 0.
3. **Create a table named orders:**
   * order\_id: Integer, Auto Increment, Primary Key.
   * item\_id: Integer, Foreign Key referencing menu\_items(item\_id).
   * order\_date: Date, Not Null.
   * quantity: Integer, Not Null, Check constraint to ensure greater than 0.
4. **Insert the following restaurants:**
   * ('Pizza Place', '123 Pizza Rd, Springfield', 4.5)
   * ('Sushi Spot', '456 Sushi Ave, Springfield', 4.8)
5. **Insert the following menu items:**
   * ('Margherita Pizza', 10.00)
   * ('California Roll', 15.00)
6. **Place an order for Margherita Pizza:**
   * Order date is 2024-09-05 and quantity is 2.
7. **Try to insert a menu item with a negative price (This should fail).**
8. **Update the rating of a restaurant to exceed 5 (This should fail).**

**Exercise 17: Event Management System**

**Exercise:**

1. **Create a table named attendees:**
   * attendee\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * phone: Variable character, Unique, Not Null.
   * registration\_date: Date, Not Null.
2. **Create a table named events:**
   * event\_id: Integer, Auto Increment, Primary Key.
   * event\_name: Variable character, Unique, Not Null.
   * event\_date: Date, Not Null.
   * max\_attendees: Integer, Not Null, Check constraint to ensure greater than 0.
3. **Create a table named registrations:**
   * registration\_id: Integer, Auto Increment, Primary Key.
   * attendee\_id: Integer, Foreign Key referencing attendees(attendee\_id).
   * event\_id: Integer, Foreign Key referencing events(event\_id).
   * registration\_time: Timestamp, Default to current timestamp.
   * Ensure that each attendee can register for the same event only once.
4. **Insert the following attendees:**
   * ('Alice Brown', 'alice.brown@example.com', '3216549870', '2024-08-15')
   * ('Bob Smith', 'bob.smith@example.com', '9876543210', '2024-08-20')
5. **Insert the following events:**
   * ('Tech Conference', '2024-09-30', 100)
   * ('Art Exhibition', '2024-10-15', 50)
6. **Register Alice for the Tech Conference.**
7. **Try to register Bob for the Tech Conference again (This should fail due to the unique constraint).**
8. **Insert an attendee with an invalid email format (This should fail).**

**Exercise 18: Online Course Platform**

**Exercise:**

1. **Create a table named instructors:**
   * instructor\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * experience\_years: Integer, Not Null, Check constraint to ensure greater than or equal to 0.
2. **Create a table named courses:**
   * course\_id: Integer, Auto Increment, Primary Key.
   * title: Variable character, Unique, Not Null.
   * instructor\_id: Integer, Foreign Key referencing instructors(instructor\_id).
   * description: Text, Not Null.
   * price: Decimal(10, 2), Not Null, Check constraint to ensure greater than 0.
3. **Create a table named enrollments:**
   * enrollment\_id: Integer, Auto Increment, Primary Key.
   * course\_id: Integer, Foreign Key referencing courses(course\_id).
   * student\_id: Integer, Foreign Key referencing students(student\_id).
   * enrollment\_date: Date, Not Null.
   * Ensure that each student can enroll in the same course only once.
4. **Insert the following instructors:**
   * ('Dr. Emily White', 'emily.white@example.com', 5)
   * ('Prof. John Green', 'john.green@example.com', 10)
5. **Insert the following courses:**
   * ('Data Science 101', 1, 'Introduction to Data Science', 199.99)
   * ('Web Development Bootcamp', 2, 'Learn to build websites', 299.99)
6. **Enroll a student (assume student\_id = 1) in Data Science 101.**
7. **Try to enroll the same student in Data Science 101 again (This should fail due to the unique constraint).**
8. **Insert an instructor with negative experience years (This should fail).**

**Exercise 19: Fitness Tracking App**

**Exercise:**

1. **Create a table named users:**
   * user\_id: Integer, Auto Increment, Primary Key.
   * username: Variable character, Unique, Not Null.
   * email: Variable character, Unique, Not Null.
   * password: Variable character, Not Null.
2. **Create a table named activities:**
   * activity\_id: Integer, Auto Increment, Primary Key.
   * activity\_name: Variable character, Unique, Not Null.
   * calories\_burned: Integer, Not Null, Check constraint to ensure greater than 0.
3. **Create a table named user\_activities:**
   * user\_activity\_id: Integer, Auto Increment, Primary Key.
   * user\_id: Integer, Foreign Key referencing users(user\_id).
   * activity\_id: Integer, Foreign Key referencing activities(activity\_id).
   * date: Date, Not Null.
   * Ensure that a user can log the same activity on the same date only once.
4. **Insert the following users:**
   * ('john\_doe', 'john.doe@example.com', 'password123')
   * ('jane\_doe', 'jane.doe@example.com', 'password456')
5. **Insert the following activities:**
   * ('Running', 500)
   * ('Cycling', 300)
6. **Log an activity for John Doe (assume user\_id = 1) for Running on 2024-09-01.**
7. **Try to log the same activity for John Doe on the same date (This should fail).**
8. **Insert an activity with negative calories burned (This should fail).**

**Exercise 20: Hotel Reservation System**

**Exercise:**

1. **Create a table named hotels:**
   * hotel\_id: Integer, Auto Increment, Primary Key.
   * hotel\_name: Variable character, Unique, Not Null.
   * location: Variable character, Not Null.
   * rating: Decimal(2, 1), Check constraint to ensure between 0 and 5.
2. **Create a table named rooms:**
   * room\_id: Integer, Auto Increment, Primary Key.
   * hotel\_id: Integer, Foreign Key referencing hotels(hotel\_id).
   * room\_type: Variable character, Not Null.
   * price\_per\_night: Decimal(10, 2), Not Null, Check constraint to ensure greater than 0.
3. **Create a table named reservations:**
   * reservation\_id: Integer, Auto Increment, Primary Key.
   * room\_id: Integer, Foreign Key referencing rooms(room\_id).
   * customer\_name: Variable character, Not Null.
   * check\_in: Date, Not Null.
   * check\_out: Date, Not Null.
   * Ensure that check-out is after check-in.
4. **Insert the following hotels:**
   * ('Grand Hotel', 'New York', 4.5)
   * ('Beach Resort', 'Miami', 4.8)
5. **Insert the following rooms:**
   * (1, 'Deluxe Suite', 300.00)
   * (2, 'Ocean View Room', 250.00)
6. **Make a reservation for a Deluxe Suite at the Grand Hotel from 2024-09-01 to 2024-09-05.**
7. **Try to make a reservation for the same room with overlapping dates (This should fail).**
8. **Insert a room with a negative price per night (This should fail).**

**Exercise 22: E-commerce System**

**Exercise:**

1. **Create a table named customers:**
   * customer\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * phone: Variable character, Not Null.
   * address: Text.
2. **Create a table named categories:**
   * category\_id: Integer, Auto Increment, Primary Key.
   * category\_name: Variable character, Unique, Not Null.
3. **Create a table named products:**
   * product\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Not Null.
   * description: Text.
   * price: Decimal(10, 2), Not Null, Check constraint to ensure greater than 0.
   * stock\_quantity: Integer, Not Null, Check constraint to ensure greater than or equal to 0.
   * category\_id: Integer, Foreign Key referencing categories(category\_id).
4. **Create a table named orders:**
   * order\_id: Integer, Auto Increment, Primary Key.
   * customer\_id: Integer, Foreign Key referencing customers(customer\_id).
   * order\_date: Date, Not Null.
   * total\_amount: Decimal(10, 2), Not Null.
5. **Create a table named order\_items:**
   * order\_item\_id: Integer, Auto Increment, Primary Key.
   * order\_id: Integer, Foreign Key referencing orders(order\_id).
   * product\_id: Integer, Foreign Key referencing products(product\_id).
   * quantity: Integer, Not Null.
6. **Create a table named payments:**
   * payment\_id: Integer, Auto Increment, Primary Key.
   * order\_id: Integer, Foreign Key referencing orders(order\_id).
   * payment\_date: Date, Not Null.
   * amount: Decimal(10, 2), Not Null.
   * payment\_method: Variable character, Not Null.
7. **Create a table named shipping:**
   * shipping\_id: Integer, Auto Increment, Primary Key.
   * order\_id: Integer, Foreign Key referencing orders(order\_id).
   * shipping\_address: Text, Not Null.
   * shipping\_date: Date.
   * delivery\_date: Date.
8. **Create a table named carts:**
   * cart\_id: Integer, Auto Increment, Primary Key.
   * customer\_id: Integer, Foreign Key referencing customers(customer\_id).
9. **Create a table named cart\_items:**
   * cart\_item\_id: Integer, Auto Increment, Primary Key.
   * cart\_id: Integer, Foreign Key referencing carts(cart\_id).
   * product\_id: Integer, Foreign Key referencing products(product\_id).
   * quantity: Integer, Not Null.
10. **Create a table named reviews:**
    * review\_id: Integer, Auto Increment, Primary Key.
    * product\_id: Integer, Foreign Key referencing products(product\_id).
    * customer\_id: Integer, Foreign Key referencing customers(customer\_id).
    * rating: Integer, Check constraint between 1 and 5.
    * comment: Text.

**Exercise 23: Hospital Management System**

**Exercise:**

1. **Create a table named patients:**
   * patient\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * dob: Date, Not Null.
   * gender: Enum ('Male', 'Female', 'Other'), Not Null.
   * phone: Variable character, Unique, Not Null.
   * email: Variable character, Unique, Not Null.
   * address: Text.
2. **Create a table named doctors:**
   * doctor\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * specialty: Variable character, Not Null.
   * phone: Variable character, Unique, Not Null.
   * email: Variable character, Unique, Not Null.
3. **Create a table named departments:**
   * department\_id: Integer, Auto Increment, Primary Key.
   * department\_name: Variable character, Unique, Not Null.
4. **Create a table named appointments:**
   * appointment\_id: Integer, Auto Increment, Primary Key.
   * patient\_id: Integer, Foreign Key referencing patients(patient\_id).
   * doctor\_id: Integer, Foreign Key referencing doctors(doctor\_id).
   * appointment\_date: DateTime, Not Null.
   * status: Enum ('Scheduled', 'Completed', 'Cancelled'), Not Null.
5. **Create a table named medical\_records:**
   * record\_id: Integer, Auto Increment, Primary Key.
   * patient\_id: Integer, Foreign Key referencing patients(patient\_id).
   * doctor\_id: Integer, Foreign Key referencing doctors(doctor\_id).
   * diagnosis: Text, Not Null.
   * treatment: Text, Not Null.
   * record\_date: Date, Not Null.
6. **Create a table named medications:**
   * medication\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Unique, Not Null.
   * description: Text.
7. **Create a table named prescriptions:**
   * prescription\_id: Integer, Auto Increment, Primary Key.
   * record\_id: Integer, Foreign Key referencing medical\_records(record\_id).
   * medication\_id: Integer, Foreign Key referencing medications(medication\_id).
   * dosage: Variable character, Not Null.
   * frequency: Variable character, Not Null.
8. **Create a table named bills:**
   * bill\_id: Integer, Auto Increment, Primary Key.
   * patient\_id: Integer, Foreign Key referencing patients(patient\_id).
   * amount: Decimal(10, 2), Not Null.
   * payment\_status: Enum ('Paid', 'Pending', 'Cancelled'), Not Null.
   * billing\_date: Date, Not Null.
9. **Create a table named staff:**
   * staff\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * position: Variable character, Not Null.
   * department\_id: Integer, Foreign Key referencing departments(department\_id).
10. **Create a table named insurance:**
    * insurance\_id: Integer, Auto Increment, Primary Key.
    * patient\_id: Integer, Foreign Key referencing patients(patient\_id).
    * provider: Variable character, Not Null.
    * policy\_number: Variable character, Unique, Not Null.
    * coverage\_amount: Decimal(10, 2), Not Null.

**Exercise 24: Library Management System**

**Exercise:**

1. **Create a table named authors:**
   * author\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * date\_of\_birth: Date, Not Null.
   * nationality: Variable character, Not Null.
2. **Create a table named publishers:**
   * publisher\_id: Integer, Auto Increment, Primary Key.
   * name: Variable character, Unique, Not Null.
   * address: Text, Not Null.
   * contact\_number: Variable character, Unique, Not Null.
3. **Create a table named categories:**
   * category\_id: Integer, Auto Increment, Primary Key.
   * category\_name: Variable character, Unique, Not Null.
4. **Create a table named books:**
   * book\_id: Integer, Auto Increment, Primary Key.
   * title: Variable character, Not Null.
   * author\_id: Integer, Foreign Key referencing authors(author\_id).
   * publisher\_id: Integer, Foreign Key referencing publishers(publisher\_id).
   * category\_id: Integer, Foreign Key referencing categories(category\_id).
   * isbn: Variable character, Unique, Not Null.
   * publication\_date: Date, Not Null.
   * copies\_available: Integer, Not Null CHECK (copies\_available >= 0).
5. **Create a table named members:**
   * member\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * phone: Variable character, Unique, Not Null.
   * join\_date: Date, Not Null.
6. **Create a table named loans:**
   * loan\_id: Integer, Auto Increment, Primary Key.
   * book\_id: Integer, Foreign Key referencing books(book\_id).
   * member\_id: Integer, Foreign Key referencing members(member\_id).
   * loan\_date: Date, Not Null.
   * return\_date: Date, Nullable.
7. **Create a table named fines:**
   * fine\_id: Integer, Auto Increment, Primary Key.
   * loan\_id: Integer, Foreign Key referencing loans(loan\_id).
   * amount: Decimal(10, 2), Not Null CHECK (amount > 0).
   * paid: Boolean, Not Null DEFAULT FALSE.
8. **Create a table named reservations:**
   * reservation\_id: Integer, Auto Increment, Primary Key.
   * book\_id: Integer, Foreign Key referencing books(book\_id).
   * member\_id: Integer, Foreign Key referencing members(member\_id).
   * reservation\_date: Date, Not Null.
   * status: Enum ('Active', 'Cancelled', 'Fulfilled'), Not Null.
9. **Create a table named reviews:**
   * review\_id: Integer, Auto Increment, Primary Key.
   * book\_id: Integer, Foreign Key referencing books(book\_id).
   * member\_id: Integer, Foreign Key referencing members(member\_id).
   * rating: Integer CHECK (rating >= 1 AND rating <= 5), Not Null.
   * comment: Text, Nullable.
10. **Create a table named transactions:**
    * transaction\_id: Integer, Auto Increment, Primary Key.
    * loan\_id: Integer, Foreign Key referencing loans(loan\_id).
    * transaction\_date: Date, Not Null.
    * amount: Decimal(10, 2), Not Null CHECK (amount > 0).

**Exercise 25: University Management System**

**Exercise:**

1. **Create a table named departments:**
   * department\_id: Integer, Auto Increment, Primary Key.
   * department\_name: Variable character, Unique, Not Null.
   * location: Variable character, Not Null.
2. **Create a table named courses:**
   * course\_id: Integer, Auto Increment, Primary Key.
   * course\_name: Variable character, Unique, Not Null.
   * department\_id: Integer, Foreign Key referencing departments(department\_id).
   * credits: Integer, Not Null CHECK (credits > 0).
3. **Create a table named students:**
   * student\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * email: Variable character, Unique, Not Null.
   * date\_of\_birth: Date, Not Null.
   * enrollment\_date: Date, Not Null.
4. **Create a table named professors:**
   * professor\_id: Integer, Auto Increment, Primary Key.
   * first\_name: Variable character, Not Null.
   * last\_name: Variable character, Not Null.
   * department\_id: Integer, Foreign Key referencing departments(department\_id).
   * hire\_date: Date, Not Null.
5. **Create a table named enrollments:**
   * enrollment\_id: Integer, Auto Increment, Primary Key.
   * student\_id: Integer, Foreign Key referencing students(student\_id).
   * course\_id: Integer, Foreign Key referencing courses(course\_id).
   * semester: Variable character, Not Null.
   * year: Integer, Not Null CHECK (year > 2000).
6. **Create a table named grades:**
   * grade\_id: Integer, Auto Increment, Primary Key.
   * enrollment\_id: Integer, Foreign Key referencing enrollments(enrollment\_id).
   * grade: Char(1) CHECK (grade IN ('A', 'B', 'C', 'D', 'F')), Not Null.
7. **Create a table named assignments:**
   * assignment\_id: Integer, Auto Increment, Primary Key.
   * course\_id: Integer, Foreign Key referencing courses(course\_id).
   * title: Variable character, Not Null.
   * due\_date: Date, Not Null.
8. **Create a table named submissions:**
   * submission\_id: Integer, Auto Increment, Primary Key.
   * assignment\_id: Integer, Foreign Key referencing assignments(assignment\_id).
   * student\_id: Integer, Foreign Key referencing students(student\_id).
   * submission\_date: Date, Not Null.
   * score: Decimal(5, 2) CHECK (score >= 0 AND score <= 100), Nullable.
9. **Create a table named classrooms:**
   * classroom\_id: Integer, Auto Increment, Primary Key.
   * room\_number: Variable character, Unique, Not Null.
   * capacity: Integer, Not Null CHECK (capacity > 0).
10. **Create a table named class\_schedule:**
    * schedule\_id: Integer, Auto Increment, Primary Key.
    * course\_id: Integer, Foreign Key referencing courses(course\_id).
    * classroom\_id: Integer, Foreign Key referencing classrooms(classroom\_id).
    * day\_of\_week: Enum ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'), Not Null.
    * start\_time: Time, Not Null.
    * end\_time: Time, Not Null CHECK (end\_time > start\_time).