**Case Study 1: Library Management System**

**Scenario:**  
A library wants to keep track of its books, members, and issued books.

**Table Structures**

**books Table**

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

genre VARCHAR(50),

publication\_year YEAR,

copies\_available INT DEFAULT 0 CHECK (copies\_available >= 0)

);

**members Table**

CREATE TABLE members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(15) UNIQUE,

join\_date DATE DEFAULT CURDATE()

);

**issued\_books Table**

CREATE TABLE issued\_books (

issue\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

member\_id INT,

issue\_date DATE DEFAULT CURDATE(),

return\_date DATE,

FOREIGN KEY (book\_id) REFERENCES books(book\_id),

FOREIGN KEY (member\_id) REFERENCES members(member\_id)

);

**Solution Queries**

1. **Issue a book to a member**
2. INSERT INTO issued\_books (book\_id, member\_id, return\_date)
3. VALUES (1, 2, '2024-02-15');
4. **Update available copies after issuing a book**
5. UPDATE books
6. SET copies\_available = copies\_available - 1
7. WHERE book\_id = 1;
8. **Get all books issued by a member**
9. SELECT b.title, b.author, i.issue\_date, i.return\_date
10. FROM issued\_books i
11. JOIN books b ON i.book\_id = b.book\_id
12. WHERE i.member\_id = 2;

**Case Study 2: Online Shopping System**

**Scenario:**  
An e-commerce store needs to manage products, customers, orders, and order details.

**Table Structures**

**products Table**

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

category VARCHAR(50),

price DECIMAL(10,2) CHECK (price > 0),

stock INT CHECK (stock >= 0)

);

**customers Table**

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

address TEXT

);

**orders Table**

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATETIME DEFAULT NOW(),

status ENUM('Pending', 'Shipped', 'Delivered') DEFAULT 'Pending',

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

**order\_details Table**

CREATE TABLE order\_details (

order\_detail\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT CHECK (quantity > 0),

subtotal DECIMAL(10,2) GENERATED ALWAYS AS (quantity \* (SELECT price FROM products WHERE products.product\_id = order\_details.product\_id)) STORED,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

**Solution Queries**

1. **Insert a new order**
2. INSERT INTO orders (customer\_id, status) VALUES (1, 'Pending');
3. **Insert order details and reduce product stock**
4. INSERT INTO order\_details (order\_id, product\_id, quantity)
5. VALUES (1, 3, 2);
6. UPDATE products
7. SET stock = stock - 2
8. WHERE product\_id = 3;
9. **Get total sales per product**
10. SELECT p.name, SUM(od.quantity) AS total\_sold, SUM(od.subtotal) AS total\_revenue
11. FROM order\_details od
12. JOIN products p ON od.product\_id = p.product\_id
13. GROUP BY p.name;

**Case Study 3: Hospital Management System**

**Scenario:**  
A hospital wants to manage doctors, patients, and appointments.

**Table Structures**

**doctors Table**

CREATE TABLE doctors (

doctor\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

specialty VARCHAR(100),

phone VARCHAR(15) UNIQUE NOT NULL

);

**patients Table**

CREATE TABLE patients (

patient\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

dob DATE,

phone VARCHAR(15) UNIQUE NOT NULL

);

**appointments Table**

CREATE TABLE appointments (

appointment\_id INT AUTO\_INCREMENT PRIMARY KEY,

doctor\_id INT,

patient\_id INT,

appointment\_date DATETIME,

status ENUM('Scheduled', 'Completed', 'Cancelled') DEFAULT 'Scheduled',

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id),

FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id)

);

**Solution Queries**

1. **Schedule an appointment**
2. INSERT INTO appointments (doctor\_id, patient\_id, appointment\_date, status)
3. VALUES (2, 3, '2024-02-05 10:00:00', 'Scheduled');
4. **Get upcoming appointments for a doctor**
5. SELECT p.name, a.appointment\_date, a.status
6. FROM appointments a
7. JOIN patients p ON a.patient\_id = p.patient\_id
8. WHERE a.doctor\_id = 2 AND a.appointment\_date > NOW();
9. **Count total appointments per doctor**
10. SELECT d.name, COUNT(a.appointment\_id) AS total\_appointments
11. FROM doctors d
12. JOIN appointments a ON d.doctor\_id = a.doctor\_id
13. GROUP BY d.name;

**Case Study 4: School Management System**

**Scenario:**  
A school needs to track students, teachers, and class enrollments.

**Table Structures**

**students Table**

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

dob DATE,

class VARCHAR(50)

);

**teachers Table**

CREATE TABLE teachers (

teacher\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

subject VARCHAR(100)

);

**enrollments Table**

CREATE TABLE enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

teacher\_id INT,

enrollment\_date DATE DEFAULT CURDATE(),

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (teacher\_id) REFERENCES teachers(teacher\_id)

);

**Solution Queries**

1. **Enroll a student in a class**
2. INSERT INTO enrollments (student\_id, teacher\_id) VALUES (3, 1);
3. **Find students taught by a specific teacher**
4. SELECT s.name
5. FROM enrollments e
6. JOIN students s ON e.student\_id = s.student\_id
7. WHERE e.teacher\_id = 1;

**Case Study 5: Employee Payroll System**

**Scenario:**  
A company needs to track employee salaries and payments.

**Table Structures**

**employees Table**

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

position VARCHAR(100),

salary DECIMAL(10,2) CHECK (salary > 0)

);

**payments Table**

CREATE TABLE payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

employee\_id INT,

payment\_date DATE DEFAULT CURDATE(),

amount DECIMAL(10,2),

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

**Solution Queries**

1. **Process a salary payment**
2. INSERT INTO payments (employee\_id, amount)
3. SELECT employee\_id, salary FROM employees WHERE employee\_id = 1;
4. **Get total payments made to employees**
5. SELECT e.name, SUM(p.amount) AS total\_paid
6. FROM payments p
7. JOIN employees e ON p.employee\_id = e.employee\_id
8. GROUP BY e.name;