**Scenario 1: Employee Management System**

A company wants to store employee details, their departments, and salaries.

**Table Structures**

CREATE TABLE Department (

dept\_id INT PRIMARY KEY AUTO\_INCREMENT,

dept\_name VARCHAR(100) NOT NULL UNIQUE

);

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY AUTO\_INCREMENT,

emp\_name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL,

dept\_id INT,

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

FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id) ON DELETE SET NULL

);

**Queries**

1. **Insert Data**

INSERT INTO Department (dept\_name) VALUES ('HR'), ('IT'), ('Finance');

INSERT INTO Employee (emp\_name, email, dept\_id, salary)

VALUES ('John Doe', 'john.doe@example.com', 2, 75000),

('Jane Smith', 'jane.smith@example.com', 1, 65000),

('Mark Brown', 'mark.brown@example.com', 3, 85000);

1. **Retrieve Employee Details Along with Department Name**

SELECT e.emp\_id, e.emp\_name, e.email, d.dept\_name, e.salary

FROM Employee e

LEFT JOIN Department d ON e.dept\_id = d.dept\_id;

1. **Update Salary of Employees in IT Department by 10%**

UPDATE Employee

SET salary = salary \* 1.10

WHERE dept\_id = (SELECT dept\_id FROM Department WHERE dept\_name = 'IT');

**Scenario 2: E-commerce Order System**

A company needs a system to track customers, products, and their orders.

**Table Structures**

CREATE TABLE Customer (

cust\_id INT PRIMARY KEY AUTO\_INCREMENT,

cust\_name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL

);

CREATE TABLE Product (

prod\_id INT PRIMARY KEY AUTO\_INCREMENT,

prod\_name VARCHAR(100) NOT NULL,

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

);

CREATE TABLE Orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

cust\_id INT,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (cust\_id) REFERENCES Customer(cust\_id) ON DELETE CASCADE

);

CREATE TABLE Order\_Details (

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

order\_id INT,

prod\_id INT,

quantity INT CHECK (quantity > 0),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id) ON DELETE CASCADE,

FOREIGN KEY (prod\_id) REFERENCES Product(prod\_id) ON DELETE CASCADE

);

**Queries**

1. **Insert Data**

INSERT INTO Customer (cust\_name, email) VALUES ('Alice', 'alice@example.com'), ('Bob', 'bob@example.com');

INSERT INTO Product (prod\_name, price) VALUES ('Laptop', 1000), ('Mouse', 25);

INSERT INTO Orders (cust\_id) VALUES (1), (2);

INSERT INTO Order\_Details (order\_id, prod\_id, quantity) VALUES (1, 1, 2), (1, 2, 3), (2, 1, 1);

1. **Retrieve Customer Orders**

SELECT o.order\_id, c.cust\_name, p.prod\_name, od.quantity, p.price, (od.quantity \* p.price) AS total\_price

FROM Orders o

JOIN Customer c ON o.cust\_id = c.cust\_id

JOIN Order\_Details od ON o.order\_id = od.order\_id

JOIN Product p ON od.prod\_id = p.prod\_id;

1. **Find the Total Amount Spent by Each Customer**

SELECT c.cust\_name, SUM(od.quantity \* p.price) AS total\_spent

FROM Orders o

JOIN Customer c ON o.cust\_id = c.cust\_id

JOIN Order\_Details od ON o.order\_id = od.order\_id

JOIN Product p ON od.prod\_id = p.prod\_id

GROUP BY c.cust\_name;

**Scenario 3: Library Management System**

A library needs to manage books, members, and book borrowings.

**Table Structures**

CREATE TABLE Members (

member\_id INT PRIMARY KEY AUTO\_INCREMENT,

member\_name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL

);

CREATE TABLE Books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(200) NOT NULL,

author VARCHAR(100),

available\_copies INT CHECK (available\_copies >= 0)

);

CREATE TABLE Borrowed\_Books (

borrow\_id INT PRIMARY KEY AUTO\_INCREMENT,

member\_id INT,

book\_id INT,

borrow\_date DATE DEFAULT CURDATE(),

return\_date DATE NULL,

FOREIGN KEY (member\_id) REFERENCES Members(member\_id) ON DELETE CASCADE,

FOREIGN KEY (book\_id) REFERENCES Books(book\_id) ON DELETE CASCADE

);

**Queries**

1. **Insert Data**

INSERT INTO Members (member\_name, email) VALUES ('John Smith', 'john.smith@example.com'), ('Emma Davis', 'emma.davis@example.com');

INSERT INTO Books (title, author, available\_copies) VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', 5), ('1984', 'George Orwell', 3);

INSERT INTO Borrowed\_Books (member\_id, book\_id) VALUES (1, 1), (2, 2);

1. **Find Currently Borrowed Books**

SELECT m.member\_name, b.title, bb.borrow\_date

FROM Borrowed\_Books bb

JOIN Members m ON bb.member\_id = m.member\_id

JOIN Books b ON bb.book\_id = b.book\_id

WHERE bb.return\_date IS NULL;

1. **Mark a Book as Returned**

UPDATE Borrowed\_Books

SET return\_date = CURDATE()

WHERE member\_id = 1 AND book\_id = 1;

**Scenario 4: Hospital Management System**

A hospital needs to store patient details, doctors, and appointments.

**Table Structures**

CREATE TABLE Doctor (

doctor\_id INT PRIMARY KEY AUTO\_INCREMENT,

doctor\_name VARCHAR(100) NOT NULL,

specialization VARCHAR(100)

);

CREATE TABLE Patient (

patient\_id INT PRIMARY KEY AUTO\_INCREMENT,

patient\_name VARCHAR(100) NOT NULL,

contact VARCHAR(15) UNIQUE NOT NULL

);

CREATE TABLE Appointment (

appointment\_id INT PRIMARY KEY AUTO\_INCREMENT,

patient\_id INT,

doctor\_id INT,

appointment\_date DATETIME NOT NULL,

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

FOREIGN KEY (patient\_id) REFERENCES Patient(patient\_id) ON DELETE CASCADE,

FOREIGN KEY (doctor\_id) REFERENCES Doctor(doctor\_id) ON DELETE CASCADE

);

**Queries**

1. **Insert Data**

INSERT INTO Doctor (doctor\_name, specialization) VALUES ('Dr. Smith', 'Cardiology'), ('Dr. Brown', 'Neurology');

INSERT INTO Patient (patient\_name, contact) VALUES ('Michael Green', '1234567890'), ('Sophia White', '0987654321');

INSERT INTO Appointment (patient\_id, doctor\_id, appointment\_date) VALUES (1, 1, '2024-02-10 10:00:00'), (2, 2, '2024-02-11 11:00:00');

1. **Get Upcoming Appointments**

SELECT a.appointment\_id, p.patient\_name, d.doctor\_name, a.appointment\_date, a.status

FROM Appointment a

JOIN Patient p ON a.patient\_id = p.patient\_id

JOIN Doctor d ON a.doctor\_id = d.doctor\_id

WHERE a.appointment\_date > NOW();

1. **Cancel an Appointment**

UPDATE Appointment

SET status = 'Cancelled'

WHERE appointment\_id = 1;