**Exercise 1: Basic INSERT Operation**

1. Create a table books with columns:
   * book\_id (INT, AUTO\_INCREMENT, Primary Key)
   * title (VARCHAR(100))
   * author (VARCHAR(50))
   * price (DECIMAL(10,2))
2. Insert 5 books with different titles, authors, and prices into the table.

**Solution**:

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author VARCHAR(50),

price DECIMAL(10,2)

);

INSERT INTO books (title, author, price)

VALUES

('The Alchemist', 'Paulo Coelho', 9.99),

('To Kill a Mockingbird', 'Harper Lee', 7.49),

('1984', 'George Orwell', 12.99),

('The Great Gatsby', 'F. Scott Fitzgerald', 10.99),

('Moby Dick', 'Herman Melville', 8.99);

**Exercise 2: Updating Data in a Table**

1. Create a table employees with columns:
   * employee\_id (INT, AUTO\_INCREMENT, Primary Key)
   * first\_name (VARCHAR(50))
   * last\_name (VARCHAR(50))
   * salary (DECIMAL(10,2))
2. Insert 4 employees with salaries into the table.
3. Update the salary of employees with a last name of 'Doe' to 60000.

**Solution**:

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

salary DECIMAL(10,2)

);

INSERT INTO employees (first\_name, last\_name, salary)

VALUES

('John', 'Doe', 50000),

('Jane', 'Smith', 55000),

('Mike', 'Doe', 50000),

('Alice', 'Johnson', 62000);

-- Update salary for 'Doe' employees

UPDATE employees

SET salary = 60000

WHERE last\_name = 'Doe';

**Exercise 3: Deleting Records**

1. Create a table orders with columns:
   * order\_id (INT, AUTO\_INCREMENT, Primary Key)
   * order\_date (DATE)
   * customer\_id (INT)
2. Insert 5 orders with different dates and customer IDs.
3. Delete all orders where the customer\_id is 3.

**Solution**:

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_date DATE,

customer\_id INT

);

INSERT INTO orders (order\_date, customer\_id)

VALUES

('2024-01-01', 1),

('2024-01-05', 2),

('2024-01-10', 3),

('2024-01-15', 3),

('2024-01-20', 4);

-- Delete orders with customer\_id = 3

DELETE FROM orders

WHERE customer\_id = 3;

**Exercise 4: Inserting Data with Foreign Keys**

1. Create two tables categories and products:
   * categories:
     + category\_id (INT, AUTO\_INCREMENT, Primary Key)
     + category\_name (VARCHAR(50))
   * products:
     + product\_id (INT, AUTO\_INCREMENT, Primary Key)
     + product\_name (VARCHAR(100))
     + price (DECIMAL(10,2))
     + category\_id (INT, Foreign Key referencing categories.category\_id)
2. Insert 3 categories into categories.
3. Insert 5 products into products linked to different categories.

**Solution**:

CREATE TABLE categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(50)

);

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2),

category\_id INT,

FOREIGN KEY (category\_id) REFERENCES categories(category\_id)

);

-- Insert categories

INSERT INTO categories (category\_name)

VALUES

('Electronics'),

('Books'),

('Clothing');

-- Insert products

INSERT INTO products (product\_name, price, category\_id)

VALUES

('Smartphone', 699.99, 1),

('Laptop', 999.99, 1),

('Novel', 19.99, 2),

('T-Shirt', 12.99, 3),

('Jeans', 49.99, 3);

**Exercise 5: Updating Data Across Multiple Tables**

1. Using the products and categories tables from the previous exercise, update the price of all products under the category 'Electronics' by increasing it by 10%.

**Solution**:

-- Increase the price of products in the 'Electronics' category by 10%

UPDATE products

SET price = price \* 1.10

WHERE category\_id = (SELECT category\_id FROM categories WHERE category\_name = 'Electronics');

**Exercise 6: Deleting Related Records with Foreign Keys**

1. Using the products and categories tables, delete all products under the 'Clothing' category.

**Solution**:

-- Delete all products in the 'Clothing' category

DELETE FROM products

WHERE category\_id = (SELECT category\_id FROM categories WHERE category\_name = 'Clothing');

**Exercise 7: Inserting Multiple Rows**

1. Create a table students with columns:
   * student\_id (INT, AUTO\_INCREMENT, Primary Key)
   * student\_name (VARCHAR(100))
   * grade (INT)
2. Insert 10 students with random names and grades into the table.

**Solution**:

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

grade INT

);

-- Insert 10 students

INSERT INTO students (student\_name, grade)

VALUES

('John Doe', 85),

('Jane Smith', 92),

('Alice Johnson', 78),

('Bob Brown', 88),

('Charlie Davis', 90),

('Daniel Adams', 80),

('Emily Clark', 75),

('Frank White', 93),

('Grace Lee', 84),

('Hannah Scott', 89);

**Exercise 8: Updating and Deleting Complex Records**

1. Create a table payments with columns:
   * payment\_id (INT, AUTO\_INCREMENT, Primary Key)
   * amount (DECIMAL(10,2))
   * payment\_date (DATE)
   * customer\_id (INT)
2. Insert 5 payments into the table.
3. Update the amount of payments made by customer\_id = 2 by increasing it by 15%.
4. Delete payments made on '2024-01-01'.

**Solution**:

CREATE TABLE payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

amount DECIMAL(10,2),

payment\_date DATE,

customer\_id INT

);

-- Insert payments

INSERT INTO payments (amount, payment\_date, customer\_id)

VALUES

(100.00, '2024-01-01', 1),

(150.00, '2024-01-05', 2),

(200.00, '2024-01-10', 3),

(300.00, '2024-01-01', 4),

(250.00, '2024-01-20', 2);

-- Update payments for customer\_id = 2

UPDATE payments

SET amount = amount \* 1.15

WHERE customer\_id = 2;

-- Delete payments made on '2024-01-01'

DELETE FROM payments

WHERE payment\_date = '2024-01-01';

**Exercise 9: Multiple INSERTs and Complex Conditions in UPDATE**

1. Create the following tables:
   * departments:
     + dept\_id (INT, AUTO\_INCREMENT, Primary Key)
     + dept\_name (VARCHAR(50))
     + location (VARCHAR(50))
   * employees:
     + emp\_id (INT, AUTO\_INCREMENT, Primary Key)
     + emp\_name (VARCHAR(100))
     + salary (DECIMAL(10,2))
     + dept\_id (INT, Foreign Key referencing departments.dept\_id)
2. Insert at least 3 departments.
3. Insert at least 10 employees.
4. Update the salary of all employees who work in the 'IT' department, increasing it by 20%.

**Solution**:

-- Create departments table

CREATE TABLE departments (

dept\_id INT AUTO\_INCREMENT PRIMARY KEY,

dept\_name VARCHAR(50),

location VARCHAR(50)

);

-- Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_name VARCHAR(100),

salary DECIMAL(10,2),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

-- Insert departments

INSERT INTO departments (dept\_name, location)

VALUES

('HR', 'New York'),

('IT', 'San Francisco'),

('Sales', 'Chicago');

-- Insert employees

INSERT INTO employees (emp\_name, salary, dept\_id)

VALUES

('John Doe', 60000, 1),

('Jane Smith', 75000, 2),

('Alice Johnson', 50000, 3),

('Bob Brown', 80000, 2),

('Charlie Davis', 67000, 1),

('Daniel Adams', 92000, 2),

('Emily Clark', 56000, 3),

('Frank White', 85000, 2),

('Grace Lee', 59000, 1),

('Hannah Scott', 64000, 3);

-- Update salary for employees in the IT department

UPDATE employees

SET salary = salary \* 1.20

WHERE dept\_id = 2;

**Exercise 10: DELETE Operation with Multiple Tables**

1. Create two tables:
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + city (VARCHAR(50))
   * orders:
     + order\_id (INT, AUTO\_INCREMENT, Primary Key)
     + order\_date (DATE)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
2. Insert at least 5 customers and 10 orders.
3. Delete all customers who are from 'Los Angeles' and delete their related orders.

**Solution**:

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

city VARCHAR(50)

);

-- Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_date DATE,

customer\_id INT,

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

);

-- Insert customers

INSERT INTO customers (customer\_name, city)

VALUES

('Michael Brown', 'New York'),

('Sarah Davis', 'Los Angeles'),

('John Doe', 'Chicago'),

('Emily Taylor', 'Los Angeles'),

('James Wilson', 'San Francisco');

-- Insert orders

INSERT INTO orders (order\_date, customer\_id)

VALUES

('2024-01-10', 1),

('2024-01-12', 2),

('2024-01-15', 3),

('2024-01-18', 4),

('2024-01-20', 5),

('2024-01-25', 1),

('2024-01-28', 2),

('2024-02-01', 3),

('2024-02-05', 4),

('2024-02-10', 5);

-- Delete customers from 'Los Angeles' and their related orders

DELETE FROM orders

WHERE customer\_id IN (SELECT customer\_id FROM customers WHERE city = 'Los Angeles');

DELETE FROM customers

WHERE city = 'Los Angeles';

**Exercise 11: Inserting and Updating Data Using Complex Conditions**

1. Create a table projects:
   * project\_id (INT, AUTO\_INCREMENT, Primary Key)
   * project\_name (VARCHAR(100))
   * start\_date (DATE)
   * end\_date (DATE)
   * budget (DECIMAL(15,2))
2. Insert at least 5 projects with different dates and budgets.
3. Update the budget of all projects that started after '2024-01-01' and have an end date before '2024-12-31', increasing the budget by 15%.

**Solution**:

-- Create projects table

CREATE TABLE projects (

project\_id INT AUTO\_INCREMENT PRIMARY KEY,

project\_name VARCHAR(100),

start\_date DATE,

end\_date DATE,

budget DECIMAL(15,2)

);

-- Insert projects

INSERT INTO projects (project\_name, start\_date, end\_date, budget)

VALUES

('Project A', '2024-01-10', '2024-06-15', 100000.00),

('Project B', '2024-03-01', '2024-08-20', 250000.00),

('Project C', '2024-04-12', '2024-10-01', 175000.00),

('Project D', '2023-12-01', '2024-05-30', 300000.00),

('Project E', '2024-07-15', '2024-12-20', 500000.00);

-- Update budget for projects started after '2024-01-01' and ending before '2024-12-31'

UPDATE projects

SET budget = budget \* 1.15

WHERE start\_date > '2024-01-01' AND end\_date < '2024-12-31';

**Exercise 12: Complex DELETE with Conditions**

1. Create the following tables:
   * students:
     + student\_id (INT, AUTO\_INCREMENT, Primary Key)
     + student\_name (VARCHAR(100))
     + enrollment\_year (YEAR)
     + major (VARCHAR(50))
   * courses:
     + course\_id (INT, AUTO\_INCREMENT, Primary Key)
     + course\_name (VARCHAR(100))
     + credit\_hours (INT)
2. Insert 10 students and 5 courses.
3. Delete all students who enrolled before 2020 and are majoring in 'Mathematics'.

**Solution**:

-- Create students table

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

enrollment\_year YEAR,

major VARCHAR(50)

);

-- Create courses table

CREATE TABLE courses (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100),

credit\_hours INT

);

-- Insert students

INSERT INTO students (student\_name, enrollment\_year, major)

VALUES

('Alice Johnson', 2019, 'Mathematics'),

('Bob Brown', 2020, 'Computer Science'),

('Charlie Davis', 2018, 'Physics'),

('Daniel Adams', 2021, 'Mathematics'),

('Emily Clark', 2019, 'Mathematics'),

('Frank White', 2020, 'Physics'),

('Grace Lee', 2021, 'Computer Science'),

('Hannah Scott', 2019, 'History'),

('John Doe', 2021, 'Mathematics'),

('Jane Smith', 2020, 'Physics');

-- Insert courses

INSERT INTO courses (course\_name, credit\_hours)

VALUES

('Calculus', 3),

('Physics I', 4),

('History of Art', 3),

('Programming Basics', 5),

('Advanced Algorithms', 4);

-- Delete students who enrolled before 2020 and major in Mathematics

DELETE FROM students

WHERE enrollment\_year < 2020 AND major = 'Mathematics';

**Exercise 13: Multiple Updates Across Tables**

1. Create the following tables:
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + city (VARCHAR(50))
   * orders:
     + order\_id (INT, AUTO\_INCREMENT, Primary Key)
     + order\_date (DATE)
     + total\_amount (DECIMAL(10,2))
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
2. Insert 5 customers and 10 orders.
3. Increase the total amount of all orders placed by customers in 'New York' by 20%.

**Solution**:

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

city VARCHAR(50)

);

-- Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_date DATE,

total\_amount DECIMAL(10,2),

customer\_id INT,

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

);

-- Insert customers

INSERT INTO customers (customer\_name, city)

VALUES

('John Doe', 'New York'),

('Jane Smith', 'Los Angeles'),

('Emily Clark', 'Chicago'),

('Michael Brown', 'New York'),

('Sarah Davis', 'San Francisco');

-- Insert orders

INSERT INTO orders (order\_date, total\_amount, customer\_id)

VALUES

('2024-01-10', 150.00, 1),

('2024-01-12', 300.00, 2),

('2024-01-15', 250.00, 3),

('2024-01-18', 400.00, 4),

('2024-01-20', 500.00, 5),

('2024-01-25', 100.00, 1),

('2024-01-28', 600.00, 2),

('2024-02-01', 200.00, 3),

('2024-02-05', 700.00, 4),

('2024-02-10', 800.00, 5);

-- Update total amount for orders placed by customers in New York

UPDATE orders

SET total\_amount = total\_amount \* 1.20

WHERE customer\_id IN (SELECT customer\_id FROM customers WHERE city = 'New York');

**Exercise 14: Managing Product Inventory**

1. Create the following tables:
   * suppliers:
     + supplier\_id (INT, AUTO\_INCREMENT, Primary Key)
     + supplier\_name (VARCHAR(100))
     + contact\_number (VARCHAR(15))
   * products:
     + product\_id (INT, AUTO\_INCREMENT, Primary Key)
     + product\_name (VARCHAR(100))
     + price (DECIMAL(10,2))
     + stock\_quantity (INT)
     + supplier\_id (INT, Foreign Key referencing suppliers.supplier\_id)
2. Insert at least 4 suppliers and 10 products.
3. Update the stock quantity of all products supplied by a supplier whose name contains 'Tech', increasing the stock by 50 units.

**Solution**:

-- Create suppliers table

CREATE TABLE suppliers (

supplier\_id INT AUTO\_INCREMENT PRIMARY KEY,

supplier\_name VARCHAR(100),

contact\_number VARCHAR(15)

);

-- Create products table

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2),

stock\_quantity INT,

supplier\_id INT,

FOREIGN KEY (supplier\_id) REFERENCES suppliers(supplier\_id)

);

-- Insert suppliers

INSERT INTO suppliers (supplier\_name, contact\_number)

VALUES

('Tech Supplies', '123-456-7890'),

('Office World', '987-654-3210'),

('Home Essentials', '555-555-5555'),

('Tech Gadgets', '444-444-4444');

-- Insert products

INSERT INTO products (product\_name, price, stock\_quantity, supplier\_id)

VALUES

('Laptop', 999.99, 20, 1),

('Office Chair', 89.99, 100, 2),

('Printer', 150.00, 50, 1),

('Notebook', 5.00, 200, 2),

('Desk Lamp', 25.00, 150, 3),

('USB Cable', 10.00, 300, 1),

('Desk', 200.00, 80, 2),

('Mouse', 30.00, 200, 4),

('Keyboard', 50.00, 120, 4),

('Monitor', 300.00, 60, 1);

-- Update stock quantity for products supplied by 'Tech' suppliers

UPDATE products

SET stock\_quantity = stock\_quantity + 50

WHERE supplier\_id IN (SELECT supplier\_id FROM suppliers WHERE supplier\_name LIKE '%Tech%');

**Exercise 15: Managing Library Records**

1. Create the following tables:
   * authors:
     + author\_id (INT, AUTO\_INCREMENT, Primary Key)
     + author\_name (VARCHAR(100))
     + nationality (VARCHAR(50))
   * books:
     + book\_id (INT, AUTO\_INCREMENT, Primary Key)
     + title (VARCHAR(100))
     + publication\_year (YEAR)
     + author\_id (INT, Foreign Key referencing authors.author\_id)
2. Insert at least 5 authors and 15 books.
3. Update the publication year of all books written by authors from 'USA' to the current year (2024).

**Solution**:

-- Create authors table

CREATE TABLE authors (

author\_id INT AUTO\_INCREMENT PRIMARY KEY,

author\_name VARCHAR(100),

nationality VARCHAR(50)

);

-- Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

publication\_year YEAR,

author\_id INT,

FOREIGN KEY (author\_id) REFERENCES authors(author\_id)

);

-- Insert authors

INSERT INTO authors (author\_name, nationality)

VALUES

('John Smith', 'USA'),

('Maria Garcia', 'Mexico'),

('Takeshi Yamamoto', 'Japan'),

('Emily Johnson', 'USA'),

('Ali Khan', 'Pakistan');

-- Insert books

INSERT INTO books (title, publication\_year, author\_id)

VALUES

('Learning SQL', 2020, 1),

('Advanced Programming', 2021, 1),

('Introduction to Databases', 2019, 2),

('Japanese Culture', 2023, 3),

('Modern Literature', 2022, 4),

('Traveling the World', 2018, 5),

('Database Design', 2021, 1),

('Cultural Insights', 2024, 3),

('Programming Basics', 2020, 2),

('History of the USA', 2024, 4),

('Cooking 101', 2022, 5),

('Technology Trends', 2023, 1),

('Web Development', 2021, 2),

('Data Science Essentials', 2024, 3),

('The Art of Programming', 2024, 4),

('Travel Guide', 2023, 5);

-- Update publication year for books by USA authors

UPDATE books

SET publication\_year = 2024

WHERE author\_id IN (SELECT author\_id FROM authors WHERE nationality = 'USA');

**Exercise 16: Deleting Unused Data in Sales Records**

1. Create the following tables:
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + email (VARCHAR(100))
   * sales:
     + sale\_id (INT, AUTO\_INCREMENT, Primary Key)
     + sale\_date (DATE)
     + amount (DECIMAL(10,2))
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
2. Insert at least 5 customers and 20 sales records.
3. Delete all sales records associated with customers who have a non-valid email format (i.e., emails without '@').

**Solution**:

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

email VARCHAR(100)

);

-- Create sales table

CREATE TABLE sales (

sale\_id INT AUTO\_INCREMENT PRIMARY KEY,

sale\_date DATE,

amount DECIMAL(10,2),

customer\_id INT,

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

);

-- Insert customers

INSERT INTO customers (customer\_name, email)

VALUES

('Alice Johnson', 'alice@example.com'),

('Bob Brown', 'bob.brown.com'),

('Charlie Davis', 'charlie@example.com'),

('Daniel Adams', 'danieladams@example'),

('Emily Clark', 'emily@example.com');

-- Insert sales records

INSERT INTO sales (sale\_date, amount, customer\_id)

VALUES

('2024-01-01', 150.00, 1),

('2024-01-02', 200.00, 1),

('2024-01-05', 100.00, 2),

('2024-01-10', 300.00, 3),

('2024-01-15', 250.00, 4),

('2024-01-20', 50.00, 1),

('2024-01-25', 80.00, 2),

('2024-01-28', 400.00, 3),

('2024-02-01', 150.00, 5),

('2024-02-05', 300.00, 4),

('2024-02-10', 250.00, 1),

('2024-02-15', 500.00, 2),

('2024-02-20', 600.00, 3),

('2024-02-25', 700.00, 5),

('2024-02-28', 800.00, 1),

('2024-03-01', 900.00, 2),

('2024-03-05', 1000.00, 3),

('2024-03-10', 1100.00, 4),

('2024-03-15', 1200.00, 5),

('2024-03-20', 1300.00, 1);

-- Delete sales records for customers with invalid email formats

DELETE FROM sales

WHERE customer\_id IN (SELECT customer\_id FROM customers WHERE email NOT LIKE '%@%');

**Exercise 17: Updating Employee Salaries Based on Conditions**

1. Create the following tables:
   * departments:
     + dept\_id (INT, AUTO\_INCREMENT, Primary Key)
     + dept\_name (VARCHAR(50))
   * employees:
     + emp\_id (INT, AUTO\_INCREMENT, Primary Key)
     + emp\_name (VARCHAR(100))
     + salary (DECIMAL(10,2))
     + dept\_id (INT, Foreign Key referencing departments.dept\_id)
2. Insert at least 4 departments and 12 employees.
3. Increase the salary of all employees in the 'HR' department by 10% and decrease the salary of all employees in the 'IT' department by 5%.

**Solution**:

-- Create departments table

CREATE TABLE departments (

dept\_id INT AUTO\_INCREMENT PRIMARY KEY,

dept\_name VARCHAR(50)

);

-- Create employees table

CREATE TABLE employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_name VARCHAR(100),

salary DECIMAL(10,2),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES departments(dept\_id)

);

-- Insert departments

INSERT INTO departments (dept\_name)

VALUES

('HR'),

('IT'),

('Sales'),

('Marketing');

-- Insert employees

INSERT INTO employees (emp\_name, salary, dept\_id)

VALUES

('Alice Smith', 60000, 1),

('Bob Johnson', 70000, 2),

('Charlie Brown', 80000, 2),

('David Wilson', 55000, 1),

('Emily Davis', 75000, 3),

('Frank Miller', 90000, 3),

('Grace Lee', 62000, 1),

('Hannah Garcia', 72000, 2),

('Isabella Martinez', 67000, 4),

('Jack Anderson', 56000, 4),

('Kelly Thompson', 74000, 3),

('Lucas White', 59000, 1);

-- Update salaries for HR department

UPDATE employees

SET salary = salary \* 1.10

WHERE dept\_id = (SELECT dept\_id FROM departments WHERE dept\_name = 'HR');

-- Update salaries for IT department

UPDATE employees

SET salary = salary \* 0.95

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

**Exercise 18: Hotel Booking Management**

1. Create the following tables:
   * guests:
     + guest\_id (INT, AUTO\_INCREMENT, Primary Key)
     + guest\_name (VARCHAR(100))
     + phone\_number (VARCHAR(15))
     + email (VARCHAR(100))
   * rooms:
     + room\_id (INT, AUTO\_INCREMENT, Primary Key)
     + room\_number (VARCHAR(10))
     + room\_type (VARCHAR(50))
     + price\_per\_night (DECIMAL(10,2))
   * bookings:
     + booking\_id (INT, AUTO\_INCREMENT, Primary Key)
     + guest\_id (INT, Foreign Key referencing guests.guest\_id)
     + room\_id (INT, Foreign Key referencing rooms.room\_id)
     + check\_in\_date (DATE)
     + check\_out\_date (DATE)
2. Insert at least 5 guests, 10 rooms, and 5 bookings.
3. Update the phone number for all guests who have a booking with room type 'Deluxe' to a new phone number.
4. Delete all bookings that have a check-out date older than '2024-01-01'.

**Solution**:

-- Create guests table

CREATE TABLE guests (

guest\_id INT AUTO\_INCREMENT PRIMARY KEY,

guest\_name VARCHAR(100),

phone\_number VARCHAR(15),

email VARCHAR(100)

);

-- Create rooms table

CREATE TABLE rooms (

room\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_number VARCHAR(10),

room\_type VARCHAR(50),

price\_per\_night DECIMAL(10,2)

);

-- Create bookings table

CREATE TABLE bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

guest\_id INT,

room\_id INT,

check\_in\_date DATE,

check\_out\_date DATE,

FOREIGN KEY (guest\_id) REFERENCES guests(guest\_id),

FOREIGN KEY (room\_id) REFERENCES rooms(room\_id)

);

-- Insert guests

INSERT INTO guests (guest\_name, phone\_number, email)

VALUES

('John Doe', '123-456-7890', 'john.doe@example.com'),

('Jane Smith', '234-567-8901', 'jane.smith@example.com'),

('Michael Johnson', '345-678-9012', 'michael.j@example.com'),

('Emily Davis', '456-789-0123', 'emily.davis@example.com'),

('Sarah Brown', '567-890-1234', 'sarah.brown@example.com');

-- Insert rooms

INSERT INTO rooms (room\_number, room\_type, price\_per\_night)

VALUES

('101', 'Standard', 100.00),

('102', 'Deluxe', 200.00),

('103', 'Suite', 300.00),

('104', 'Deluxe', 250.00),

('105', 'Standard', 120.00),

('201', 'Deluxe', 220.00),

('202', 'Suite', 350.00),

('203', 'Standard', 90.00),

('204', 'Suite', 400.00),

('205', 'Deluxe', 210.00);

-- Insert bookings

INSERT INTO bookings (guest\_id, room\_id, check\_in\_date, check\_out\_date)

VALUES

(1, 2, '2024-01-01', '2024-01-05'),

(2, 4, '2024-01-10', '2024-01-15'),

(3, 1, '2024-01-20', '2024-01-25'),

(4, 2, '2024-02-01', '2024-02-05'),

(5, 5, '2024-01-28', '2024-02-02');

-- Update phone numbers for guests with Deluxe room bookings

UPDATE guests

SET phone\_number = '999-999-9999'

WHERE guest\_id IN (SELECT guest\_id FROM bookings WHERE room\_id IN (SELECT room\_id FROM rooms WHERE room\_type = 'Deluxe'));

-- Delete bookings with check-out date older than '2024-01-01'

DELETE FROM bookings

WHERE check\_out\_date < '2024-01-01';

**Exercise 19: University Course Enrollment**

1. Create the following tables:
   * students:
     + student\_id (INT, AUTO\_INCREMENT, Primary Key)
     + student\_name (VARCHAR(100))
     + major (VARCHAR(50))
   * courses:
     + course\_id (INT, AUTO\_INCREMENT, Primary Key)
     + course\_name (VARCHAR(100))
     + credits (INT)
   * enrollments:
     + enrollment\_id (INT, AUTO\_INCREMENT, Primary Key)
     + student\_id (INT, Foreign Key referencing students.student\_id)
     + course\_id (INT, Foreign Key referencing courses.course\_id)
2. Insert at least 5 students, 8 courses, and 10 enrollments.
3. Update the major of all students who are enrolled in a course with more than 3 credits to 'Undeclared'.
4. Delete enrollments for courses that are less than 3 credits.

**Solution**:

-- Create students table

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

major VARCHAR(50)

);

-- Create courses table

CREATE TABLE courses (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100),

credits INT

);

-- Create enrollments table

CREATE TABLE enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

course\_id INT,

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

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

);

-- Insert students

INSERT INTO students (student\_name, major)

VALUES

('Alice Green', 'Computer Science'),

('Bob White', 'Mathematics'),

('Charlie Black', 'Biology'),

('David Grey', 'Chemistry'),

('Eve Red', 'Physics');

-- Insert courses

INSERT INTO courses (course\_name, credits)

VALUES

('Database Systems', 4),

('Calculus', 3),

('Biochemistry', 2),

('Linear Algebra', 3),

('Physics I', 4),

('Chemistry II', 3),

('Web Development', 3),

('Artificial Intelligence', 4);

-- Insert enrollments

INSERT INTO enrollments (student\_id, course\_id)

VALUES

(1, 1),

(1, 5),

(2, 2),

(2, 3),

(3, 1),

(3, 4),

(4, 2),

(4, 5),

(5, 3),

(5, 6);

-- Update major for students enrolled in courses with more than 3 credits

UPDATE students

SET major = 'Undeclared'

WHERE student\_id IN (SELECT student\_id FROM enrollments WHERE course\_id IN (SELECT course\_id FROM courses WHERE credits > 3));

-- Delete enrollments for courses with less than 3 credits

DELETE FROM enrollments

WHERE course\_id IN (SELECT course\_id FROM courses WHERE credits < 3);

**Exercise 20: E-Commerce Order Management**

1. Create the following tables:
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + email (VARCHAR(100))
   * products:
     + product\_id (INT, AUTO\_INCREMENT, Primary Key)
     + product\_name (VARCHAR(100))
     + price (DECIMAL(10,2))
   * orders:
     + order\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + order\_date (DATE)
     + total\_amount (DECIMAL(10,2))
2. Insert at least 5 customers, 8 products, and 10 orders.
3. Update the total amount of all orders placed by customers with a specific email domain to reflect a 10% discount.
4. Delete all products that have never been ordered.

**Solution**:

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

email VARCHAR(100)

);

-- Create products table

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2)

);

-- Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10,2),

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

);

-- Insert customers

INSERT INTO customers (customer\_name, email)

VALUES

('Alice Johnson', 'alice@example.com'),

('Bob Smith', 'bob@example.com'),

('Charlie Brown', 'charlie@example.org'),

('David Wilson', 'david@example.com'),

('Emily Clark', 'emily@example.org');

-- Insert products

INSERT INTO products (product\_name, price)

VALUES

('Laptop', 1200.00),

('Smartphone', 800.00),

('Tablet', 500.00),

('Headphones', 150.00),

('Smartwatch', 300.00),

('Camera', 600.00),

('Printer', 200.00),

('Monitor', 400.00);

-- Insert orders

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES

(1, '2024-01-01', 1300.00),

(1, '2024-01-02', 1000.00),

(2, '2024-01-10', 500.00),

(3, '2024-01-15', 800.00),

(4, '2024-02-01', 200.00),

(5, '2024-02-02', 750.00),

(1, '2024-02-10', 400.00),

(2, '2024-02-15', 300.00),

(3, '2024-03-01', 600.00),

(4, '2024-03-05', 150.00);

-- Update total amount for orders placed by customers with email domain 'example.com'

UPDATE orders

SET total\_amount = total\_amount \* 0.90

WHERE customer\_id IN (SELECT customer\_id FROM customers WHERE email LIKE '%@example.com');

-- Delete products that have never been ordered

DELETE FROM products

WHERE product\_id NOT IN (SELECT DISTINCT product\_id FROM orders);

**Exercise 21: Library Management System**

1. **Create the following tables**:
   * members:
     + member\_id (INT, AUTO\_INCREMENT, Primary Key)
     + member\_name (VARCHAR(100))
     + email (VARCHAR(100))
     + membership\_date (DATE)
   * books:
     + book\_id (INT, AUTO\_INCREMENT, Primary Key)
     + book\_title (VARCHAR(200))
     + author (VARCHAR(100))
     + published\_year (YEAR)
     + available\_copies (INT)
   * loans:
     + loan\_id (INT, AUTO\_INCREMENT, Primary Key)
     + member\_id (INT, Foreign Key referencing members.member\_id)
     + book\_id (INT, Foreign Key referencing books.book\_id)
     + loan\_date (DATE)
     + return\_date (DATE)
2. **Insert at least**:
   * 5 members
   * 10 books
   * 15 loans
3. **Perform the following operations**:
   * Update the available\_copies for all books where the author is 'John Smith' by reducing it by 1.
   * Delete loans that have a return\_date older than '2023-01-01'.
   * Update the email of members who joined before '2023-01-01' to a new email format (e.g., append .old).

**Solution**:

-- Create members table

CREATE TABLE members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

member\_name VARCHAR(100),

email VARCHAR(100),

membership\_date DATE

);

-- Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_title VARCHAR(200),

author VARCHAR(100),

published\_year YEAR,

available\_copies INT

);

-- Create loans table

CREATE TABLE loans (

loan\_id INT AUTO\_INCREMENT PRIMARY KEY,

member\_id INT,

book\_id INT,

loan\_date DATE,

return\_date DATE,

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

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

);

-- Insert members

INSERT INTO members (member\_name, email, membership\_date)

VALUES

('Alice Johnson', 'alice@example.com', '2021-01-15'),

('Bob Smith', 'bob@example.com', '2020-05-10'),

('Charlie Brown', 'charlie@example.com', '2022-02-20'),

('David Wilson', 'david@example.com', '2019-08-25'),

('Emily Clark', 'emily@example.com', '2023-03-01');

-- Insert books

INSERT INTO books (book\_title, author, published\_year, available\_copies)

VALUES

('Learning SQL', 'John Smith', 2020, 5),

('Data Structures', 'Jane Doe', 2019, 3),

('Algorithms Unlocked', 'John Smith', 2021, 4),

('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 2),

('To Kill a Mockingbird', 'Harper Lee', 1960, 3),

('1984', 'George Orwell', 1949, 4),

('Pride and Prejudice', 'Jane Austen', 1813, 5),

('The Catcher in the Rye', 'J.D. Salinger', 1951, 1),

('Moby Dick', 'Herman Melville', 1851, 2),

('War and Peace', 'Leo Tolstoy', 1869, 0);

-- Insert loans

INSERT INTO loans (member\_id, book\_id, loan\_date, return\_date)

VALUES

(1, 1, '2023-01-01', '2023-01-15'),

(1, 2, '2023-02-01', '2023-02-10'),

(2, 3, '2023-03-05', '2023-03-20'),

(2, 4, '2023-04-01', '2023-04-15'),

(3, 5, '2023-05-10', '2023-05-25'),

(4, 1, '2023-06-01', '2023-06-15'),

(5, 6, '2023-07-01', NULL),

(1, 7, '2023-08-01', '2023-08-15'),

(3, 8, '2023-09-01', NULL),

(4, 9, '2023-10-01', '2023-10-15'),

(2, 10, '2023-11-01', NULL),

(5, 2, '2023-12-01', NULL),

(1, 3, '2024-01-05', NULL),

(2, 4, '2024-01-10', NULL),

(3, 5, '2024-01-15', NULL);

-- Update available\_copies for books by John Smith

UPDATE books

SET available\_copies = available\_copies - 1

WHERE author = 'John Smith';

-- Delete loans with return\_date older than '2023-01-01'

DELETE FROM loans

WHERE return\_date < '2023-01-01';

-- Update emails for members who joined before '2023-01-01'

UPDATE members

SET email = CONCAT(SUBSTRING\_INDEX(email, '@', 1), '.old@', SUBSTRING\_INDEX(email, '@', -1))

WHERE membership\_date < '2023-01-01';

**Exercise 22: Sales Management System**

1. **Create the following tables**:
   * sales\_reps:
     + rep\_id (INT, AUTO\_INCREMENT, Primary Key)
     + rep\_name (VARCHAR(100))
     + region (VARCHAR(50))
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + contact\_number (VARCHAR(15))
   * sales:
     + sale\_id (INT, AUTO\_INCREMENT, Primary Key)
     + rep\_id (INT, Foreign Key referencing sales\_reps.rep\_id)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + sale\_amount (DECIMAL(10,2))
     + sale\_date (DATE)
2. **Insert at least**:
   * 5 sales reps
   * 10 customers
   * 20 sales records
3. **Perform the following operations**:
   * Update the region for all sales reps whose names start with 'A' to 'North'.
   * Delete sales records where the sale\_amount is less than 100.00.
   * Update the contact number for customers whose names contain 'Inc' to a new format (e.g., add area code).

**Solution**:

-- Create sales\_reps table

CREATE TABLE sales\_reps (

rep\_id INT AUTO\_INCREMENT PRIMARY KEY,

rep\_name VARCHAR(100),

region VARCHAR(50)

);

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

contact\_number VARCHAR(15)

);

-- Create sales table

CREATE TABLE sales (

sale\_id INT AUTO\_INCREMENT PRIMARY KEY,

rep\_id INT,

customer\_id INT,

sale\_amount DECIMAL(10,2),

sale\_date DATE,

FOREIGN KEY (rep\_id) REFERENCES sales\_reps(rep\_id),

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

);

-- Insert sales reps

INSERT INTO sales\_reps (rep\_name, region)

VALUES

('Alice Green', 'West'),

('Bob White', 'East'),

('Charlie Black', 'South'),

('David Brown', 'Central'),

('Eve Adams', 'South');

-- Insert customers

INSERT INTO customers (customer\_name, contact\_number)

VALUES

('Acme Corp', '123-456-7890'),

('Global Tech Inc', '234-567-8901'),

('XYZ Solutions', '345-678-9012'),

('Retail Hub', '456-789-0123'),

('ABC Industries', '567-890-1234'),

('Innovatech', '678-901-2345'),

('Tech World', '789-012-3456'),

('Data Corp', '890-123-4567'),

('Fresh Foods', '901-234-5678'),

('NextGen Ltd', '012-345-6789');

-- Insert sales records

INSERT INTO sales (rep\_id, customer\_id, sale\_amount, sale\_date)

VALUES

(1, 1, 150.00, '2023-01-01'),

(1, 2, 90.00, '2023-01-10'),

(2, 3, 120.00, '2023-01-15'),

(2, 4, 200.00, '2023-01-20'),

(3, 5, 350.00, '2023-02-01'),

(3, 6, 400.00, '2023-02-05'),

(4, 7, 60.00, '2023-02-10'),

(4, 8, 500.00, '2023-02-15'),

(5, 9, 700.00, '2023-03-01'),

(5, 10, 50.00, '2023-03-05'),

(1, 3, 300.00, '2023-03-10'),

(2, 2, 250.00, '2023-03-15'),

(3, 1, 150.00, '2023-04-01'),

(4, 2, 80.00, '2023-04-05'),

(5, 4, 900.00, '2023-04-10'),

(1, 5, 400.00, '2023-04-15'),

(2, 6, 120.00, '2023-04-20'),

(3, 7, 75.00, '2023-05-01'),

(4, 8, 150.00, '2023-05-05'),

(5, 9, 300.00, '2023-05-10');

-- Update region for sales reps starting with 'A'

UPDATE sales\_reps

SET region = 'North'

WHERE rep\_name LIKE 'A%';

-- Delete sales records with sale\_amount less than 100.00

DELETE FROM sales

WHERE sale\_amount < 100.00;

-- Update contact numbers for customers with 'Inc' in their name

UPDATE customers

SET contact\_number = CONCAT('800-', SUBSTRING(contact\_number, 1, 3))

WHERE customer\_name LIKE '%Inc%';

**Exercise 23: Restaurant Management System**

1. **Create the following tables**:
   * restaurants:
     + restaurant\_id (INT, AUTO\_INCREMENT, Primary Key)
     + restaurant\_name (VARCHAR(100))
     + location (VARCHAR(100))
   * dishes:
     + dish\_id (INT, AUTO\_INCREMENT, Primary Key)
     + restaurant\_id (INT, Foreign Key referencing restaurants.restaurant\_id)
     + dish\_name (VARCHAR(100))
     + price (DECIMAL(10,2))
   * orders:
     + order\_id (INT, AUTO\_INCREMENT, Primary Key)
     + dish\_id (INT, Foreign Key referencing dishes.dish\_id)
     + quantity (INT)
     + order\_date (DATE)
2. **Insert at least**:
   * 5 restaurants
   * 15 dishes
   * 30 orders
3. **Perform the following operations**:
   * Update the price for all dishes under a specific restaurant by increasing it by 10%.
   * Delete orders that have a quantity less than 2.
   * Update the location of all restaurants that serve dishes priced above 50.00 to 'Premium Area'.

**Solution**:

-- Create restaurants table

CREATE TABLE restaurants (

restaurant\_id INT AUTO\_INCREMENT PRIMARY KEY,

restaurant\_name VARCHAR(100),

location VARCHAR(100)

);

-- Create dishes table

CREATE TABLE dishes (

dish\_id INT AUTO\_INCREMENT PRIMARY KEY,

restaurant\_id INT,

dish\_name VARCHAR(100),

price DECIMAL(10,2),

FOREIGN KEY (restaurant\_id) REFERENCES restaurants(restaurant\_id)

);

-- Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

dish\_id INT,

quantity INT,

order\_date DATE,

FOREIGN KEY (dish\_id) REFERENCES dishes(dish\_id)

);

-- Insert restaurants

INSERT INTO restaurants (restaurant\_name, location)

VALUES

('Italian Bistro', 'Downtown'),

('Sushi Place', 'Eastside'),

('Curry House', 'Westend'),

('Burger Joint', 'Midtown'),

('Dessert Café', 'Uptown');

-- Insert dishes

INSERT INTO dishes (restaurant\_id, dish\_name, price)

VALUES

(1, 'Pasta Primavera', 15.00),

(1, 'Margherita Pizza', 12.00),

(1, 'Tiramisu', 7.50),

(2, 'California Roll', 10.00),

(2, 'Sushi Platter', 20.00),

(3, 'Butter Chicken', 12.00),

(3, 'Lamb Curry', 15.00),

(4, 'Cheeseburger', 10.00),

(4, 'French Fries', 5.00),

(5, 'Chocolate Cake', 6.00),

(5, 'Ice Cream Sundae', 8.00),

(1, 'Fettuccine Alfredo', 14.00),

(2, 'Tempura Shrimp', 12.00),

(3, 'Vegetable Biryani', 9.00),

(4, 'Chicken Nuggets', 8.00),

(5, 'Fruit Tart', 7.00);

-- Insert orders

INSERT INTO orders (dish\_id, quantity, order\_date)

VALUES

(1, 2, '2023-01-01'),

(2, 1, '2023-01-01'),

(3, 3, '2023-01-02'),

(4, 1, '2023-01-03'),

(5, 4, '2023-01-04'),

(6, 2, '2023-01-05'),

(7, 5, '2023-01-06'),

(8, 1, '2023-01-07'),

(9, 1, '2023-01-08'),

(10, 3, '2023-01-09'),

(11, 2, '2023-01-10'),

(12, 2, '2023-01-11'),

(13, 1, '2023-01-12'),

(14, 1, '2023-01-13'),

(15, 1, '2023-01-14'),

(2, 3, '2023-01-15'),

(3, 2, '2023-01-16'),

(1, 4, '2023-01-17'),

(5, 5, '2023-01-18'),

(4, 2, '2023-01-19'),

(3, 2, '2023-01-20'),

(2, 1, '2023-01-21'),

(5, 3, '2023-01-22'),

(1, 4, '2023-01-23'),

(4, 1, '2023-01-24'),

(5, 2, '2023-01-25'),

(1, 5, '2023-01-26'),

(3, 4, '2023-01-27'),

(2, 2, '2023-01-28'),

(4, 1, '2023-01-29'),

(5, 1, '2023-01-30');

-- Update prices for dishes in Italian Bistro

UPDATE dishes

SET price = price \* 1.10

WHERE restaurant\_id = 1;

-- Delete orders with quantity less than 2

DELETE FROM orders

WHERE quantity < 2;

-- Update location for restaurants serving dishes priced above 50.00

UPDATE restaurants

SET location = 'Premium Area'

WHERE restaurant\_id IN (

SELECT restaurant\_id FROM dishes WHERE price > 50.00

);

**Exercise 24: Library Management System**

1. **Create the following tables**:
   * books:
     + book\_id (INT, AUTO\_INCREMENT, Primary Key)
     + title (VARCHAR(100))
     + author (VARCHAR(100))
     + publication\_year (INT)
   * members:
     + member\_id (INT, AUTO\_INCREMENT, Primary Key)
     + member\_name (VARCHAR(100))
     + membership\_date (DATE)
   * loans:
     + loan\_id (INT, AUTO\_INCREMENT, Primary Key)
     + book\_id (INT, Foreign Key referencing books.book\_id)
     + member\_id (INT, Foreign Key referencing members.member\_id)
     + loan\_date (DATE)
     + return\_date (DATE)
2. **Insert at least**:
   * 10 books
   * 5 members
   * 15 loans
3. **Perform the following operations**:
   * Update the publication\_year for all books by a specific author.
   * Delete loans that have return\_date before a specific date.
   * Update membership\_date for all members who have loans older than a year.

**Solution**:

-- Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author VARCHAR(100),

publication\_year INT

);

-- Create members table

CREATE TABLE members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

member\_name VARCHAR(100),

membership\_date DATE

);

-- Create loans table

CREATE TABLE loans (

loan\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

member\_id INT,

loan\_date DATE,

return\_date DATE,

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

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

);

-- Insert books

INSERT INTO books (title, author, publication\_year)

VALUES

('The Great Gatsby', 'F. Scott Fitzgerald', 1925),

('1984', 'George Orwell', 1949),

('To Kill a Mockingbird', 'Harper Lee', 1960),

('Pride and Prejudice', 'Jane Austen', 1813),

('The Catcher in the Rye', 'J.D. Salinger', 1951),

('Moby Dick', 'Herman Melville', 1851),

('War and Peace', 'Leo Tolstoy', 1869),

('The Odyssey', 'Homer', -800),

('The Divine Comedy', 'Dante Alighieri', 1320),

('The Brothers Karamazov', 'Fyodor Dostoevsky', 1880);

-- Insert members

INSERT INTO members (member\_name, membership\_date)

VALUES

('Alice Johnson', '2022-01-01'),

('Bob Smith', '2021-05-15'),

('Charlie Brown', '2023-03-20'),

('David Wilson', '2020-07-30'),

('Eve Davis', '2019-11-25');

-- Insert loans

INSERT INTO loans (book\_id, member\_id, loan\_date, return\_date)

VALUES

(1, 1, '2023-01-01', '2023-01-10'),

(2, 1, '2023-01-15', '2023-01-20'),

(3, 2, '2022-12-01', '2022-12-15'),

(4, 2, '2023-01-01', NULL),

(5, 3, '2023-02-01', '2023-02-10'),

(1, 4, '2023-03-01', NULL),

(6, 5, '2023-03-15', '2023-03-20'),

(7, 1, '2023-03-20', NULL),

(8, 2, '2023-04-01', NULL),

(9, 3, '2023-04-10', NULL),

(10, 4, '2023-04-15', NULL),

(2, 5, '2023-04-20', '2023-04-25'),

(3, 1, '2023-05-01', NULL),

(4, 2, '2023-05-05', '2023-05-10'),

(5, 3, '2023-05-15', NULL),

(6, 4, '2023-05-20', '2023-05-25');

-- Update publication year for books by 'George Orwell'

UPDATE books

SET publication\_year = 1950

WHERE author = 'George Orwell';

-- Delete loans with return\_date before '2023-01-01'

DELETE FROM loans

WHERE return\_date < '2023-01-01';

-- Update membership\_date for members with loans older than a year

UPDATE members

SET membership\_date = DATE\_SUB(membership\_date, INTERVAL 1 YEAR)

WHERE member\_id IN (

SELECT DISTINCT member\_id FROM loans WHERE loan\_date < DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)

);

**Exercise 25: Online Store System**

1. **Create the following tables**:
   * products:
     + product\_id (INT, AUTO\_INCREMENT, Primary Key)
     + product\_name (VARCHAR(100))
     + price (DECIMAL(10,2))
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + email (VARCHAR(100))
   * orders:
     + order\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + order\_date (DATE)
     + total\_amount (DECIMAL(10,2))
2. **Insert at least**:
   * 10 products
   * 5 customers
   * 15 orders
3. **Perform the following operations**:
   * Update the price for a specific product.
   * Delete orders that have a total amount less than a specific value.
   * Update the email address for all customers whose names start with 'J'.

**Solution**:

-- Create products table

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2)

);

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

email VARCHAR(100)

);

-- Create orders table

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10,2),

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

);

-- Insert products

INSERT INTO products (product\_name, price)

VALUES

('Laptop', 1200.00),

('Smartphone', 800.00),

('Tablet', 400.00),

('Headphones', 150.00),

('Smartwatch', 300.00),

('Monitor', 200.00),

('Keyboard', 50.00),

('Mouse', 25.00),

('Printer', 150.00),

('Webcam', 75.00);

-- Insert customers

INSERT INTO customers (customer\_name, email)

VALUES

('Alice Johnson', 'alice@example.com'),

('Bob Smith', 'bob@example.com'),

('Charlie Brown', 'charlie@example.com'),

('David Wilson', 'david@example.com'),

('Judy White', 'judy@example.com');

-- Insert orders

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES

(1, '2023-01-01', 1200.00),

(2, '2023-01-15', 800.00),

(3, '2023-01-20', 400.00),

(4, '2023-01-25', 150.00),

(5, '2023-02-01', 300.00),

(1, '2023-02-05', 200.00),

(2, '2023-02-10', 50.00),

(3, '2023-02-15', 150.00),

(4, '2023-03-01', 600.00),

(5, '2023-03-05', 700.00),

(1, '2023-03-10', 120.00),

(2, '2023-03-15', 250.00),

(3, '2023-04-01', 150.00),

(4, '2023-04-05', 80.00),

(5, '2023-04-10', 900.00);

-- Update price for 'Smartphone'

UPDATE products

SET price = 750.00

WHERE product\_name = 'Smartphone';

-- Delete orders with total\_amount less than 100.00

DELETE FROM orders

WHERE total\_amount < 100.00;

-- Update email for customers whose names start with 'J'

UPDATE customers

SET email = CONCAT(SUBSTRING(email, 1, LOCATE('@', email) - 1), '@newdomain.com')

WHERE customer\_name LIKE 'J%';

**Exercise 26: School Management System**

1. **Create the following tables**:
   * students:
     + student\_id (INT, AUTO\_INCREMENT, Primary Key)
     + student\_name (VARCHAR(100))
     + enrollment\_date (DATE)
   * courses:
     + course\_id (INT, AUTO\_INCREMENT, Primary Key)
     + course\_name (VARCHAR(100))
     + credits (INT)
   * enrollments:
     + enrollment\_id (INT, AUTO\_INCREMENT, Primary Key)
     + student\_id (INT, Foreign Key referencing students.student\_id)
     + course\_id (INT, Foreign Key referencing courses.course\_id)
     + grade (VARCHAR(2))
2. **Insert at least**:
   * 10 students
   * 5 courses
   * 20 enrollments
3. **Perform the following operations**:
   * Update the credits for a specific course.
   * Delete enrollments for students who have not passed (grade 'F').
   * Update the enrollment date for all students who enrolled before a specific date.

**Solution**:

-- Create students table

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

enrollment\_date DATE

);

-- Create courses table

CREATE TABLE courses (

course\_id INT AUTO\_INCREMENT PRIMARY KEY,

course\_name VARCHAR(100),

credits INT

);

-- Create enrollments table

CREATE TABLE enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

course\_id INT,

grade VARCHAR(2),

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

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

);

-- Insert students

INSERT INTO students (student\_name, enrollment\_date)

VALUES

('Alice Johnson', '2022-01-01'),

('Bob Smith', '2021-05-15'),

('Charlie Brown', '2020-09-20'),

('David Wilson', '2022-03-25'),

('Eve Davis', '2023-04-01'),

('Frank White', '2021-11-15'),

('Grace Green', '2022-01-10'),

('Hank Black', '2023-02-20'),

('Ivy Grey', '2021-06-30'),

('Jack Blue', '2022-08-15');

-- Insert courses

INSERT INTO courses (course\_name, credits)

VALUES

('Mathematics', 3),

('Science', 4),

('History', 3),

('Literature', 3),

('Art', 2);

-- Insert enrollments

INSERT INTO enrollments (student\_id, course\_id, grade)

VALUES

(1, 1, 'A'),

(1, 2, 'B'),

(2, 2, 'C'),

(2, 3, 'D'),

(3, 1, 'F'),

(3, 3, 'B'),

(4, 4, 'A'),

(5, 1, 'A'),

(6, 2, 'F'),

(7, 3, 'C'),

(8, 4, 'B'),

(9, 5, 'A'),

(10, 5, 'C'),

(1, 3, 'A'),

(2, 1, 'C'),

(3, 4, 'D'),

(4, 2, 'B'),

(5, 3, 'A'),

(6, 5, 'F'),

(7, 4, 'C'),

(8, 2, 'A');

-- Update credits for 'Science'

UPDATE courses

SET credits = 5

WHERE course\_name = 'Science';

-- Delete enrollments with grade 'F'

DELETE FROM enrollments

WHERE grade = 'F';

-- Update enrollment date for students who enrolled before '2022-01-01'

UPDATE students

SET enrollment\_date = DATE\_ADD(enrollment\_date, INTERVAL 1 YEAR)

WHERE enrollment\_date < '2022-01-01';

**Exercise 27: Hospital Management System**

1. **Create the following tables**:
   * patients:
     + patient\_id (INT, AUTO\_INCREMENT, Primary Key)
     + patient\_name (VARCHAR(100))
     + admission\_date (DATE)
     + discharge\_date (DATE)
   * doctors:
     + doctor\_id (INT, AUTO\_INCREMENT, Primary Key)
     + doctor\_name (VARCHAR(100))
     + specialization (VARCHAR(100))
   * appointments:
     + appointment\_id (INT, AUTO\_INCREMENT, Primary Key)
     + patient\_id (INT, Foreign Key referencing patients.patient\_id)
     + doctor\_id (INT, Foreign Key referencing doctors.doctor\_id)
     + appointment\_date (DATE)
2. **Insert at least**:
   * 10 patients
   * 5 doctors
   * 15 appointments
3. **Perform the following operations**:
   * Update the discharge date for a specific patient.
   * Delete appointments that have not occurred yet.
   * Update the specialization for all doctors who specialize in a specific area.

**Solution**:

-- Create patients table

CREATE TABLE patients (

patient\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_name VARCHAR(100),

admission\_date DATE,

discharge\_date DATE

);

-- Create doctors table

CREATE TABLE doctors (

doctor\_id INT AUTO\_INCREMENT PRIMARY KEY,

doctor\_name VARCHAR(100),

specialization VARCHAR(100)

);

-- Create appointments table

CREATE TABLE appointments (

appointment\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT,

doctor\_id INT,

appointment\_date DATE,

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

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

);

-- Insert patients

INSERT INTO patients (patient\_name, admission\_date, discharge\_date)

VALUES

('John Doe', '2023-01-01', NULL),

('Jane Smith', '2023-01-05', NULL),

('Emily Davis', '2023-01-10', NULL),

('Michael Brown', '2023-01-15', NULL),

('Sarah Wilson', '2023-01-20', NULL),

('David Johnson', '2023-01-25', '2023-02-01'),

('Laura Green', '2023-02-05', NULL),

('James White', '2023-02-10', NULL),

('Linda Black', '2023-02-15', '2023-02-20'),

('Robert Grey', '2023-02-20', NULL);

-- Insert doctors

INSERT INTO doctors (doctor\_name, specialization)

VALUES

('Dr. Smith', 'Cardiology'),

('Dr. Adams', 'Neurology'),

('Dr. Johnson', 'Pediatrics'),

('Dr. Brown', 'Orthopedics'),

('Dr. Taylor', 'General Surgery');

-- Insert appointments

INSERT INTO appointments (patient\_id, doctor\_id, appointment\_date)

VALUES

(1, 1, '2023-01-02'),

(2, 1, '2023-01-06'),

(3, 2, '2023-01-11'),

(4, 3, '2023-01-16'),

(5, 4, '2023-01-21'),

(6, 5, '2023-01-26'),

(1, 2, '2023-02-01'),

(2, 3, '2023-02-07'),

(3, 4, '2023-02-12'),

(4, 1, '2023-02-17'),

(5, 2, '2023-02-22'),

(6, 3, '2023-02-27'),

(7, 4, '2023-03-01'),

(8, 5, '2023-03-05'),

(9, 1, '2023-03-10'),

(10, 2, '2023-03-15');

-- Update discharge date for patient 'John Doe'

UPDATE patients

SET discharge\_date = '2023-01-10'

WHERE patient\_name = 'John Doe';

-- Delete appointments that have appointment\_date in the future

DELETE FROM appointments

WHERE appointment\_date > CURDATE();

-- Update specialization for doctors specializing in 'Pediatrics'

UPDATE doctors

SET specialization = 'Child Health'

WHERE specialization = 'Pediatrics';

**Exercise 28: Vehicle Rental System**

1. **Create the following tables**:
   * vehicles:
     + vehicle\_id (INT, AUTO\_INCREMENT, Primary Key)
     + vehicle\_name (VARCHAR(100))
     + rental\_price (DECIMAL(10,2))
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + license\_number (VARCHAR(20))
   * rentals:
     + rental\_id (INT, AUTO\_INCREMENT, Primary Key)
     + vehicle\_id (INT, Foreign Key referencing vehicles.vehicle\_id)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + rental\_date (DATE)
     + return\_date (DATE)
2. **Insert at least**:
   * 10 vehicles
   * 5 customers
   * 15 rentals
3. **Perform the following operations**:
   * Update the rental price for a specific vehicle.
   * Delete rentals that have a return date before a specific date.
   * Update the license number for all customers whose name starts with 'A'.

**Solution**:

-- Create vehicles table

CREATE TABLE vehicles (

vehicle\_id INT AUTO\_INCREMENT PRIMARY KEY,

vehicle\_name VARCHAR(100),

rental\_price DECIMAL(10,2)

);

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

license\_number VARCHAR(20)

);

-- Create rentals table

CREATE TABLE rentals (

rental\_id INT AUTO\_INCREMENT PRIMARY KEY,

vehicle\_id INT,

customer\_id INT,

rental\_date DATE,

return\_date DATE,

FOREIGN KEY (vehicle\_id) REFERENCES vehicles(vehicle\_id),

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

);

-- Insert vehicles

INSERT INTO vehicles (vehicle\_name, rental\_price)

VALUES

('Toyota Camry', 50.00),

('Honda Accord', 55.00),

('Ford Mustang', 75.00),

('Chevrolet Malibu', 65.00),

('Nissan Altima', 60.00),

('BMW 3 Series', 85.00),

('Mercedes-Benz C-Class', 90.00),

('Audi A4', 80.00),

('Volkswagen Jetta', 70.00),

('Subaru Legacy', 55.00);

-- Insert customers

INSERT INTO customers (customer\_name, license\_number)

VALUES

('Alice Johnson', 'A123456'),

('Bob Smith', 'B654321'),

('Charlie Brown', 'C987654'),

('David Wilson', 'D123987'),

('Eva Green', 'E654123');

-- Insert rentals

INSERT INTO rentals (vehicle\_id, customer\_id, rental\_date, return\_date)

VALUES

(1, 1, '2023-01-01', '2023-01-05'),

(2, 1, '2023-01-10', '2023-01-15'),

(3, 2, '2023-01-20', '2023-01-25'),

(4, 3, '2023-01-30', '2023-02-02'),

(5, 4, '2023-02-01', '2023-02-06'),

(1, 5, '2023-02-05', '2023-02-10'),

(6, 1, '2023-02-15', '2023-02-20'),

(7, 2, '2023-02-25', '2023-03-02'),

(8, 3, '2023-03-05', '2023-03-10'),

(9, 4, '2023-03-15', '2023-03-20'),

(10, 5, '2023-03-25', '2023-03-30'),

(2, 1, '2023-04-01', '2023-04-05'),

(3, 2, '2023-04-10', '2023-04-15'),

(4, 3, '2023-04-20', '2023-04-25'),

(5, 4, '2023-04-30', '2023-05-05'),

(1, 5, '2023-05-10', '2023-05-15');

-- Update rental price for 'Ford Mustang'

UPDATE vehicles

SET rental\_price = 70.00

WHERE vehicle\_name = 'Ford Mustang';

-- Delete rentals with return\_date before '2023-01-01'

DELETE FROM rentals

WHERE return\_date < '2023-01-01';

-- Update license number for customers whose name starts with 'A'

UPDATE customers

SET license\_number = 'NEW\_A\_NUMBER'

WHERE customer\_name LIKE 'A%';

**Exercise 29: Bookstore Management System**

1. **Create the following tables**:
   * books:
     + book\_id (INT, AUTO\_INCREMENT, Primary Key)
     + title (VARCHAR(100))
     + author (VARCHAR(100))
     + price (DECIMAL(10,2))
   * customers:
     + customer\_id (INT, AUTO\_INCREMENT, Primary Key)
     + customer\_name (VARCHAR(100))
     + email (VARCHAR(100))
   * sales:
     + sale\_id (INT, AUTO\_INCREMENT, Primary Key)
     + book\_id (INT, Foreign Key referencing books.book\_id)
     + customer\_id (INT, Foreign Key referencing customers.customer\_id)
     + sale\_date (DATE)
     + quantity (INT)
2. **Insert at least**:
   * 10 books
   * 5 customers
   * 20 sales
3. **Perform the following operations**:
   * Update the price of a specific book.
   * Delete sales records for books that are out of stock (quantity = 0).
   * Update email addresses for customers with a specific domain.

**Solution**:

-- Create books table

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author VARCHAR(100),

price DECIMAL(10,2)

);

-- Create customers table

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

email VARCHAR(100)

);

-- Create sales table

CREATE TABLE sales (

sale\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

customer\_id INT,

sale\_date DATE,

quantity INT,

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

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

);

-- Insert books

INSERT INTO books (title, author, price)

VALUES

('The Great Gatsby', 'F. Scott Fitzgerald', 10.99),

('To Kill a Mockingbird', 'Harper Lee', 12.99),

('1984', 'George Orwell', 9.99),

('Pride and Prejudice', 'Jane Austen', 8.99),

('The Catcher in the Rye', 'J.D. Salinger', 11.99),

('Moby-Dick', 'Herman Melville', 14.99),

('War and Peace', 'Leo Tolstoy', 19.99),

('The Odyssey', 'Homer', 13.99),

('Jane Eyre', 'Charlotte Brontë', 7.99),

('The Lord of the Rings', 'J.R.R. Tolkien', 25.99);

-- Insert customers

INSERT INTO customers (customer\_name, email)

VALUES

('Alice Johnson', 'alice@example.com'),

('Bob Smith', 'bob@example.com'),

('Charlie Brown', 'charlie@example.com'),

('David Wilson', 'david@example.com'),

('Eva Green', 'eva@example.com');

-- Insert sales

INSERT INTO sales (book\_id, customer\_id, sale\_date, quantity)

VALUES

(1, 1, '2023-01-01', 2),

(2, 1, '2023-01-05', 1),

(3, 2, '2023-01-10', 5),

(4, 3, '2023-01-15', 1),

(5, 4, '2023-01-20', 3),

(6, 5, '2023-01-25', 0), -- Out of stock

(7, 1, '2023-01-30', 4),

(8, 2, '2023-02-05', 0), -- Out of stock

(9, 3, '2023-02-10', 2),

(10, 4, '2023-02-15', 1),

(1, 5, '2023-02-20', 2),

(2, 1, '2023-02-25', 3),

(3, 2, '2023-03-01', 1),

(4, 3, '2023-03-05', 2),

(5, 4, '2023-03-10', 1),

(6, 5, '2023-03-15', 2),

(7, 1, '2023-03-20', 3),

(8, 2, '2023-03-25', 1),

(9, 3, '2023-04-01', 4),

(10, 4, '2023-04-05', 2);

-- Update price of '1984'

UPDATE books

SET price = 8.49

WHERE title = '1984';

-- Delete sales records for books that are out of stock

DELETE FROM sales

WHERE quantity = 0;

-- Update email addresses for customers with 'example.com' domain

UPDATE customers

SET email = REPLACE(email, 'example.com', 'newdomain.com')

WHERE email LIKE '%@example.com';

**Exercise 30: Online Store Inventory System**

1. **Create the following tables**:
   * products:
     + product\_id (INT, AUTO\_INCREMENT, Primary Key)
     + product\_name (VARCHAR(100))
     + price (DECIMAL(10,2))
     + stock (INT)
   * suppliers:
     + supplier\_id (INT, AUTO\_INCREMENT, Primary Key)
     + supplier\_name (VARCHAR(100))
     + contact\_email (VARCHAR(100))
   * purchases:
     + purchase\_id (INT, AUTO\_INCREMENT, Primary Key)
     + product\_id (INT, Foreign Key referencing products.product\_id)
     + supplier\_id (INT, Foreign Key referencing suppliers.supplier\_id)
     + purchase\_date (DATE)
     + quantity (INT)
2. **Insert at least**:
   * 10 products
   * 5 suppliers
   * 15 purchases
3. **Perform the following operations**:
   * Update the stock for a specific product.
   * Delete purchases that are older than a specific date.
   * Update contact emails for suppliers with a specific domain.

**Solution**:

-- Create products table

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2),

stock INT

);

-- Create suppliers table

CREATE TABLE suppliers (

supplier\_id INT AUTO\_INCREMENT PRIMARY KEY,

supplier\_name VARCHAR(100),

contact\_email VARCHAR(100)

);

-- Create purchases table

CREATE TABLE purchases (

purchase\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id INT,

supplier\_id INT,

purchase\_date DATE,

quantity INT,

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

FOREIGN KEY (supplier\_id) REFERENCES suppliers(supplier\_id)

);

-- Insert products

INSERT INTO products (product\_name, price, stock)

VALUES

('Laptop', 1000.00, 50),

('Smartphone', 800.00, 100),

('Tablet', 300.00, 75),

('Monitor', 200.00, 20),

('Keyboard', 50.00, 150),

('Mouse', 25.00, 200),

('Printer', 150.00, 30),

('Scanner', 120.00, 10),

('Router', 80.00, 40),

('External Hard Drive', 150.00, 25);

-- Insert suppliers

INSERT INTO suppliers (supplier\_name, contact\_email)

VALUES

('Tech Supplies Co.', 'contact@techsupplies.com'),

('Gadgets Ltd.', 'info@gadgets.com'),

('Electronics Warehouse', 'support@electronics.com'),

('Office Supplies Inc.', 'sales@officesupplies.com'),

('Computers R Us', 'hello@computersrus.com');

-- Insert purchases

INSERT INTO purchases (product\_id, supplier\_id, purchase\_date, quantity)

VALUES

(1, 1, '2023-01-01', 10),

(2, 1, '2023-01-05', 15),

(3, 2, '2023-01-10', 20),

(4, 3, '2023-01-15', 5),

(5, 4, '2023-01-20', 50),

(6, 5, '2023-01-25', 30),

(1, 2, '2023-02-01', 10),

(2, 3, '2023-02-05', 5),

(3, 4, '2023-02-10', 15),

(4, 5, '2023-02-15', 20),

(5, 1, '2023-02-20', 25),

(6, 2, '2023-02-25', 10),

(7, 3, '2023-03-01', 5),

(8, 4, '2023-03-05', 10),

(9, 5, '2023-03-10', 5);

-- Update stock for product 'Laptop'

UPDATE products

SET stock = 60

WHERE product\_name = 'Laptop';

-- Delete purchases that are older than '2023-02-01'

DELETE FROM purchases

WHERE purchase\_date < '2023-02-01';

-- Update contact emails for suppliers with 'supplies.com' domain

UPDATE suppliers

SET contact\_email = REPLACE(contact\_email, 'supplies.com', 'newdomain.com')

WHERE contact\_email LIKE '%@supplies.com';