**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.

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

author VARCHAR(50),

price DECIMAL(10,2)

);

Insert these 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.

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

salary DECIMAL(10,2)

);

Insert these values

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

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

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

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

**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.

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_date DATE,

customer\_id INT

);

Insert these rows

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

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

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

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

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

**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.

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

('Electronics'),

('Books'),

('Clothing');

-- Insert products

('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%.

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

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

**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.

CREATE TABLE students (

student\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_name VARCHAR(100),

grade INT

);

-- Insert 10 students

('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'.

CREATE TABLE payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

amount DECIMAL(10,2),

payment\_date DATE,

customer\_id INT

);

-- Insert payments

(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);

**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%.

-- 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

('HR', 'New York'),

('IT', 'San Francisco'),

('Sales', 'Chicago');

-- Insert employees

('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);

**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.

-- 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

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

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

('John Doe', 'Chicago'),

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

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

-- Insert orders

('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);

**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%.

-- 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

('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);

**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'.

-- 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

('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

('Calculus', 3),

('Physics I', 4),

('History of Art', 3),

('Programming Basics', 5),

('Advanced Algorithms', 4);

**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%.

-- 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

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

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

('Emily Clark', 'Chicago'),

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

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

-- Insert orders

('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);

**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.

-- 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

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

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

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

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

-- Insert products

('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);

**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).

-- 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

('John Smith', 'USA'),

('Maria Garcia', 'Mexico'),

('Takeshi Yamamoto', 'Japan'),

('Emily Johnson', 'USA'),

('Ali Khan', 'Pakistan');

-- Insert books

('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);

**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 '@').

-- 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

('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

('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);

**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%.

-- 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

('HR'),

('IT'),

('Sales'),

('Marketing');

-- Insert employees

('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);

**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'.

-- 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

('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

('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

(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');

**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.

-- 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

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

('Bob White', 'Mathematics'),

('Charlie Black', 'Biology'),

('David Grey', 'Chemistry'),

('Eve Red', 'Physics');

-- Insert courses

('Database Systems', 4),

('Calculus', 3),

('Biochemistry', 2),

('Linear Algebra', 3),

('Physics I', 4),

('Chemistry II', 3),

('Web Development', 3),

('Artificial Intelligence', 4);

-- Insert enrollments

(1, 1),

(1, 5),

(2, 2),

(2, 3),

(3, 1),

(3, 4),

(4, 2),

(4, 5),

(5, 3),

(5, 6);

**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.

-- 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

('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

('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

(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);