Mysql Task

// 1. Create the ecommerce database

CREATE DATABASE ecommerce;

USE ecommerce;

// 2. Create customers table

CREATE TABLE customers (

id INT AUTO\_INCREMENT PRIMARY KEY, -- Primary key for customers

name VARCHAR(255) NOT NULL, -- Customer's name

email VARCHAR(255) NOT NULL, -- Customer's email address

address TEXT -- Customer's address

);

// Create orders table

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY, -- Primary key for orders

customer\_id INT, -- Foreign key referencing customers

order\_date DATE, -- Date the order was placed

total\_amount DECIMAL(10, 2), -- Total amount of the order

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

);

// 4. Create products table

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY, -- Primary key for products

name VARCHAR(255) NOT NULL, -- Product's name

price DECIMAL(10, 2) NOT NULL, -- Product's price

description TEXT -- Product's description

);

// 5. Insert sample data into customers table

INSERT INTO customers (name, email, address) VALUES

('Alice Smith', 'alice@example.com', '123 Maple St'),

('Bob Johnson', 'bob@example.com', '456 Oak St'),

('Charlie Brown', 'charlie@example.com', '789 Pine St');

// 6. Insert sample data into products table

INSERT INTO products (name, price, description) VALUES

('Product A', 30.00, 'Description of Product A'),

('Product B', 20.00, 'Description of Product B'),

('Product C', 40.00, 'Description of Product C');

// 7. Insert sample data into orders table

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

(1, '2024-10-10', 60.00),

(2, '2024-10-20', 40.00),

(1, '2024-11-01', 30.00);

/// Queries

// 1. Retrieve all customers who have placed an order in the last 30 days

SELECT DISTINCT c.\*

FROM customers c

JOIN orders o ON c.id = o.customer\_id

WHERE o.order\_date >= DATE\_SUB(CURDATE(), INTERVAL 30 DAY);

// 2. Get the total amount of all orders placed by each customer

SELECT c.name, SUM(o.total\_amount) AS total\_spent

FROM customers c

JOIN orders o ON c.id = o.customer\_id

GROUP BY c.id;

// 3. Update the price of Product C to 45.00

UPDATE products

SET price = 45.00

WHERE name = 'Product C';

// 4. Add a new column "discount" to the products table

ALTER TABLE products

ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0.00;

// 5. Retrieve the top 3 products with the highest price

SELECT \*

FROM products

ORDER BY price DESC

LIMIT 3;

// 6. Get the names of customers who have ordered Product A

SELECT DISTINCT c.name

FROM customers c

JOIN orders o ON c.id = o.customer\_id

JOIN order\_items oi ON o.id = oi.order\_id

JOIN products p ON oi.product\_id = p.id

WHERE p.name = 'Product A';

// 7. Join the orders and customers tables to retrieve the customer's name and order date for each order

SELECT c.name, o.order\_date

FROM customers c

JOIN orders o ON c.id = o.customer\_id;

//8. Retrieve the orders with a total amount grAeater than 150.00

SELECT \*

FROM orders

WHERE total\_amount > 150.00;

// 9. Normalize the database by creating a separate table for order items

-- Step 1: Create the order\_items table to represent each product in an order

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY, -- Primary key for order items

order\_id INT, -- Foreign key referencing orders

product\_id INT, -- Foreign key referencing products

quantity INT NOT NULL, -- Quantity of the product in the order

price DECIMAL(10, 2), -- Price of the product at the time of order

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

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

);

-- Step 2: Remove the total\_amount column from orders table (since it will be calculated from order\_items)

ALTER TABLE orders

DROP COLUMN total\_amount;

// 10. Retrieve the average total of all orders

SELECT AVG(SUM(oi.price \* oi.quantity)) AS average\_order\_total

FROM orders o

JOIN order\_items oi ON o.id = oi.order\_id

GROUP BY o.id;