```
mysql> CREATE DATABASE ecommerce;
Query OK, 1 row affected (0.01 sec)

mysql> USE ecommerce;
Database changed

mysql> CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name
VARCHAR(100) NOT NULL, email VARCHAR(100) NOT NULL, address VARCHAR(255));
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE orders (id INT AUTO_INCREMENT PRIMARY KEY, customer_id INT,
order_date DATE, total_amount DECIMAL(10, 2), FOREIGN KEY (customer_id)
REFERENCES customers(id));
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE products (id INT AUTO_INCREMENT PRIMARY KEY, name
VARCHAR(100) NOT NULL, price DECIMAL(10, 2), description TEXT);
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO customers (name, email, address) VALUES ('John Doe',
'john@example.com', '123 Elm St'),('Jane Smith', 'jane@example.com', '456 Maple
Ave'),('Bob Johnson',   'bob@example.com', '789 Oak Blvd');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> 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');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO orders (customer_id, order_date, total_amount) VALUES (1,
CURDATE() - INTERVAL 15 DAY, 70.00),(2, CURDATE() - INTERVAL 5 DAY, 50.00),(3,
CURDATE() - INTERVAL 35 DAY,  90.00);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

-- Queries

-- 1. Retrieve all customers who have placed an order in the last 30 days
mysql> SELECT DISTINCT c.nameFROM customers cJOIN orders o ON c.id =
o.customer_idWHERE o.order_date >= CURDATE() - INTERVAL 30 DAY;
+------------+
| name       |
+------------+
| John Doe   |
| Jane Smith |
+------------+
2 rows in set (0.00 sec)

-- 2. Get the total amount of all orders placed by each customer
mysql> SELECT c.name, SUM(o.total_amount) AS total_spentFROM customers cJOIN
orders o ON c.id = o.customer_idGROUP BY c.name;
+-------------+-------------+
| name        | total_spent |
+-------------+-------------+
| John Doe    |       70.00 |
| Jane Smith  |       50.00 |
| Bob Johnson |       90.00 |
+-------------+-------------+
3 rows in set (0.00 sec)

-- 3. Update the price of Product C to 45.00
```

```
mysql> UPDATE products SET price = 45.00 WHERE name = 'Product C';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

-- 4. Add a new column discount to the products table
mysql> ALTER TABLE products ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0.00;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

-- 5. Retrieve the top 3 products with the highest price
mysql> SELECT name, price FROM products ORDER BY price DESC LIMIT 3;
+-----------+-------+
| name      | price |
+-----------+-------+
| Product C | 45.00 |
| Product A | 30.00 |
| Product B | 20.00 |
+-----------+-------+
3 rows in set (0.00 sec)

-- 7. Join the orders and customers tables to retrieve the customer's name and
order date for each order
mysql> SELECT c.name AS customer_name, o.order_date FROM customers c JOIN orders
o ON c.id = o.customer_id;
+---------------+------------+
| customer_name | order_date |
+---------------+------------+
| John Doe      | 2024-10-29 |
| Jane Smith    | 2024-11-08 |
| Bob Johnson   | 2024-10-09 |
+---------------+------------+
3 rows in set (0.00 sec)

-- 8. Retrieve the orders with a total amount greater than 150.00
mysql> SELECT id AS order_id, customer_id, total_amount FROM orders WHERE
total_amount > 150.00;
Empty set (0.00 sec)

-- 9. Normalize the database by creating a separate table for order items and
updating the orders table to reference the order_items table
mysql> CREATE TABLE order_items (id INT AUTO_INCREMENT PRIMARY KEY,order_id INT,
product_id INT, quantity INT DEFAULT 1, price DECIMAL(10, 2), FOREIGN KEY
(order_id) REFERENCES orders(id),
      FOREIGN KEY (product_id) REFERENCES products(id));
Query OK, 0 rows affected (0.03 sec)

-- Remove product reference from orders and use order_items instead
mysql> ALTER TABLE orders DROP COLUMN total_amount;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

-- 10. Retrieve the average total of all orders
mysql> SELECT AVG(total_amount) AS average_order_total FROM ( SELECT o.id,
SUM(oi.price * oi.quantity) AS total_amount FROM orders o JOIN order_items oi ON
o.id = oi.order_id GROUP BY o.id
      ) AS order_totals;

+---------------------+
| average_order_total |
+---------------------+
|                NULL |
+---------------------+
1 row in set (0.01 sec)
```