SQL Task:

Create a database named ecommerce.
create database ecommerce; use ecommerce;

Create three tables: customers, orders, and products.
1)Customers Table:

create table customers(id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL,email VARCHAR(100) NOT NULL,address VARCHAR(255));

2. Orders Table:

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));

3. Products Table:

create table products(id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100) NOT NULL,price DECIMAL(10,2),description TEXT);

Insert sample data:

Sample data for Customers Table:

```
insert into customers(name, email, address) values( ('John Doe',
'john.doe@example.com', '123 Elm St'),
('Jane Smith', 'jane.smith@example.com', '456 Maple Ave'), ('Alice Johnson',
'alice.j@example.com', '789 Oak Dr'),
);
```

Sample data for orders:

```
Insert into orders (customer_id, order_date, total_amount) VALUES (1,
CURDATE(), 60.00), (2, CURDATE() - INTERVAL 15 DAY, 75.00), (1, CURDATE() -
INTERVAL 35 DAY, 80.00);
```

Sample data for products Table:

```
INSERT INTO products (name, price, description) VALUES ('Product A', 20.00,
'Description of Product A'), ('Product B', 35.00, 'Description of Product B'),
('Product C', 50.00, 'Description of Product C');
```

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 >= 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 discount decimal(5,2) DEFAULT 0.00;
```

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

We have to add more products in product table using insert query,

```
Insert into products (name, price, description) values('Product D', 60.00, 'Description of
Product D'), ('Product E', 50.00, 'Description of Product E'),
('Product F', 90.00, 'Description of Product F');
```

Now, 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.

Lets, create order_items Table,
create table order_items (id INT AUTO_INCREMENT PRIMARY KEY, order_id INT, product_id INT,
quantity INT DEFAULT 1,
FOREIGN KEY (order_id) REFERENCES orders(id), FOREIGN KEY (product_id) REFERENCES products(id));

Insert Sample Data into order_items,
Insert into order_items (order_id, product_id, quantity) values (1, 1, 2),(1, 2, 1), (2, 1, 1),(3, 3, 3);

Retrieve Customer Names 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 AS customer_name, o.order_date FROM orders o JOIN customers c ON o.customer_id = c.id;
```

8. Retrieve the orders with a total amount greater than 150.00:

```
select * from orders where total_amount > 150.00;
```

9. Normalize the database by creating a separate table for order items and updating the orders table to reference the order_items table:

```
create table order_items ( id INT AUTO_INCREMENT PRIMARY KEY, order_id INT,
product_id INT, quantity INT,
FOREIGN KEY (order_id) REFERENCES orders(id),
FOREIGN KEY (product_id) REFERENCES products(id) );
```

10. Retrieve the average total of all orders:

```
select AVG(total_amount) AS average_order_total FROM orders;
```