ECOMM

Schema:

**Creation of Database:**

Create database ecomm;

Use ecomm;

**Customer Table:**

create table customers(

customer\_id int primary key,

name varchar(20) not null,

email varchar(20) not null,

address varchar(50) not null,

password varchar(30) not null

);

**Products Table:**

create table products(

product\_id int primary key,

name varchar(30) not null,

price decimal(10,3) not null,

description varchar(30) not null,

stockQuantity int not null

);

**Cart Table;**

CREATE TABLE cart (

cart\_id INT PRIMARY KEY,

customer\_id INT,

product\_id INT,

quantity INT,

CONSTRAINT cart\_fk\_cus\_id

FOREIGN KEY (customer\_id)

REFERENCES customers(customer\_id),

CONSTRAINT cart\_fk\_prod\_id

FOREIGN KEY (product\_id)

REFERENCES products(product\_id)

);

**Orders Table:**

create table orders(

order\_id int primary key,

customer\_id int ,

order\_date date not null,

total\_price decimal(10,2) not null,

shipping\_address varchar(30) not null,

constraint orders\_fk\_cus\_id

foreign key(customer\_id)

references customers(customer\_id)

);

**Order\_items\_table:**

create table order\_items(

order\_item\_id int primary key,

order\_id int,

product\_id int,

quantity int not null,

constraint order\_items\_fk\_ord\_id

foreign key(order\_id)

references orders(order\_id),

constraint order\_items\_fk\_prod\_id

foreign key(product\_id)

references products(product\_id)

);

**Table Inputs:**

**Customer Table:**

Insert into customers

(customer\_id, name, email, address, password)

values

(1, 'John Doe', 'johndoe@example.com', '123 Main St, City','JohnDoe123'),

(2, 'Jane Smith', 'janesmith@example.com', '456 Elm St, Town', 'JaneSmith123'),

(3, 'Robert Johnson', 'robert@example.com', '789 Oak St,Village', 'RobertJohnson123'),

(4, 'Sarah Brown', 'sarah@example.com', '101 Pine St, Suburb', 'SarahBrown123'),

(5, 'David Lee', 'david@example.com', '234 Cedar St, District', 'DavidLee123'),

(6, 'Laura Hall', 'laura@example.com', '567 Birch St, County', 'LauraHall123'),

(7, 'Michael Davis', 'michael@example.com', '890 Maple St, State', 'MichaelDavis123'),

(8, 'Emma Wilson', 'emma@example.com', '321 Redwood St, Country', 'EmmaWilson123'),

(9, 'William Taylor', 'william@example.com', '432 Spruce St, Province', 'WilliamTaylor123'),

(10, 'Olivia Adams', 'olivia@example.com', '765 Fir St, Territory', 'OliviaAdams123');



**Products table:**

insert into products (product\_id, name, description, price, stockQuantity)

values

(1, 'Laptop', 'High-performance laptop', 800.00, 10),

(2, 'Smartphone', 'Latest smartphone', 600.00, 15),

(3, 'Tablet', 'Portable tablet', 300.00, 20),

(4, 'Headphones', 'Noise-canceling', 150.00, 30),

(5, 'TV', '4K Smart TV', 900.00, 5),

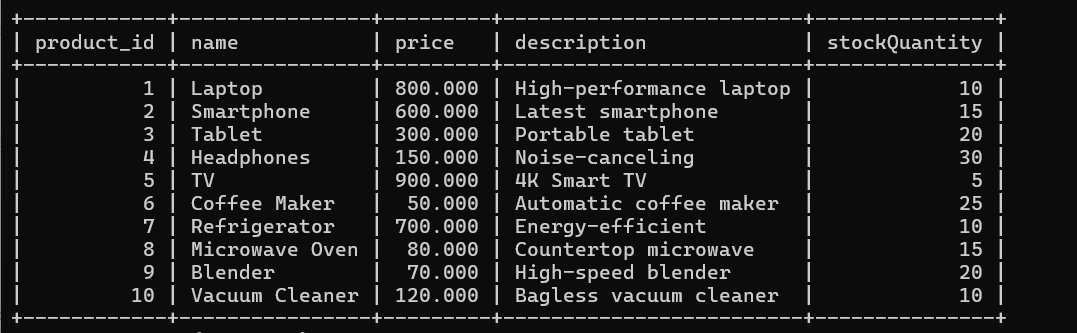
(6, 'Coffee Maker', 'Automatic coffee maker', 50.00, 25),

(7, 'Refrigerator', 'Energy-efficient', 700.00, 10),

(8, 'Microwave Oven', 'Countertop microwave', 80.00, 15),

(9, 'Blender', 'High-speed blender', 70.00, 20),

(10, 'Vacuum Cleaner', 'Bagless vacuum cleaner', 120.00, 10);



**Cart table:**

insert into cart (cart\_id, customer\_id, product\_id, quantity)

values

(1, 1, 1, 2),

(2, 1, 3, 1),

(3, 2, 2, 3),

(4, 3, 4, 4),

(5, 3, 5, 2),

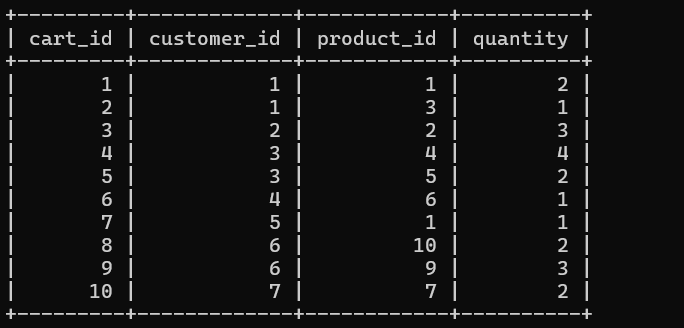
(6, 4, 6, 1),

(7, 5, 1, 1),

(8, 6, 10, 2),

(9, 6, 9, 3),

(10, 7, 7, 2);



**Orders table:**

insert into orders (order\_id, customer\_id, order\_date, total\_price, shipping\_address)

values

(1, 1, '2023-01-05', 1200.00, '123 Main St, City'),

(2, 2, '2023-02-10', 900.00, '456 Elm St, Town'),

(3, 3, '2023-03-15', 300.00, '789 Oak St, Village'),

(4, 4, '2023-04-20', 150.00, '101 Pine St, Suburb'),

(5, 5, '2023-05-25', 1800.00, '234 Cedar St, District'),

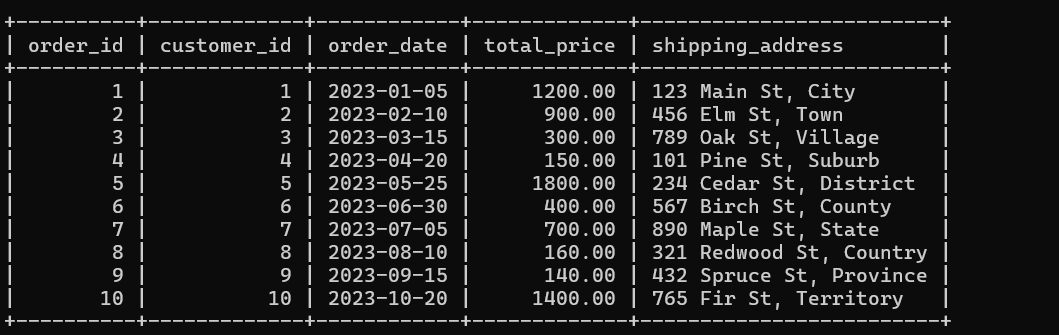
(6, 6, '2023-06-30', 400.00, '567 Birch St, County'),

(7, 7, '2023-07-05', 700.00, '890 Maple St, State'),

(8, 8, '2023-08-10', 160.00, '321 Redwood St, Country'),

(9, 9, '2023-09-15', 140.00, '432 Spruce St, Province'),

(10, 10, '2023-10-20', 1400.00, '765 Fir St, Territory');



**Order\_items:**

insert into order\_items (order\_item\_id, order\_id, product\_id,

quantity)

values

(1, 1, 1, 2),

(2, 1, 3, 1),

(3, 2, 2, 3),

(4, 3, 5, 2),

(5, 4, 4, 4),

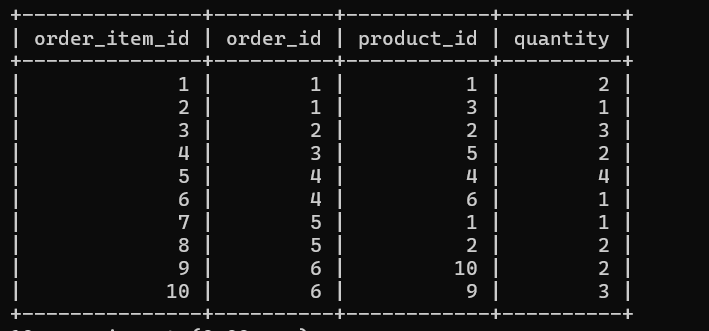
(6, 4, 6, 1),

(7, 5, 1, 1),

(8, 5, 2, 2),

(9, 6, 10, 2),

(10, 6, 9, 3);



**Task Query:**

1. Update refrigerator product price to 800.

update products

set price=800

where name="refrigerator";

2. Remove all cart items for a specific customer.

delete from cart

where customer\_id=

(select customer\_id from customers

where name="Michael Davis" );

3. Retrieve Products Priced Below $100.

select product\_id,name,price,description,stockQuantity

from products

where price < 100;

4. Find Products with Stock Quantity Greater Than 5.

select product\_id,name,stockQuantity

from products

where stockQuantity > 5;

5. Retrieve Orders with Total Amount Between $500 and $1000.

select order\_id,customer\_id,order\_date,total\_price

from orders

where total\_price between 500 and 1000;

6. Find Products which name end with letter ‘r’.

select name

from products

where name like '%r';

7. Retrieve Cart Items for Customer 5.

select cart\_id,customer\_id,product\_id,quantity

from cart

where customer\_id = 5;

8. Find Customers Who Placed Orders in 2023.

select o.customer\_id ,c.name,o.order\_date

from orders o

join customers c

on o.customer\_id=c.customer\_id

where year(order\_date)='2023'

9. Determine the Minimum Stock Quantity for Each Product Category.

select category as product\_category ,min(stockQuantity)

from products

group by category;

10. Calculate the Total Amount Spent by Each Customer.

select o.customer\_id,c.name,ifnull(o.total\_price,0) as total\_amount

from orders o

join customers c

on o.customer\_id=c.customer\_id;

11. Find the Average Order Amount for Each Customer.

select customer\_id,avg(total\_price) as average\_amount

from orders

group by customer\_id;

12. Count the Number of Orders Placed by Each Customer.

select customer\_id,count(customer\_id) as no\_of\_orders

from orders

group by customer\_id;

13. Find the Maximum Order Amount for Each Customer.

select customer\_id,max(total\_price) as max\_order\_amount

from orders

group by customer\_id;

14. Get Customers Who Placed Orders Totaling Over $1000

select customer\_id,sum(total\_price) as Total\_amount\_spent

from orders

group by customer\_id

having Total\_amount\_spent > 1000;

15. Subquery to Find Products Not in the Cart.

select product\_id,name

from products

where product\_id not in (

select product\_id from cart);

16. Subquery to Find Customers Who Haven't Placed Orders.

select customer\_id, name

from customers

where customer\_id not in (

select customer\_id from orders)

17. Subquery to Calculate the Percentage of Total Revenue for a Product.

18. Subquery to Find Products with Low Stock.

(I)

select product\_id,name ,stockQuantity

from products

where stockQuantity =

( select min(stockQuantity) from products)

(II)

select product\_id,name ,stockQuantity

from products

where product\_id in

( select product\_id from products

where stockQuantity < 15)

19. Subquery to Find Customers Who Placed High-Value Orders

select customer\_id, name

from customers

where customer\_id in (

select customer\_id

from orders

where total\_price > 1000

);