# Schema-

create database Ecom;

use Ecom;

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

description TEXT,

stockQuantity INT NOT NULL

);

CREATE TABLE cart (

cart\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

product\_id INT,

quantity INT NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE CASCADE

);

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE NOT NULL,

total\_price DECIMAL(10, 2) NOT NULL,

shipping\_address VARCHAR(255) NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE

);

CREATE TABLE order\_items (

order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE CASCADE

# );

DML Commands:-

INSERT INTO customers (customer\_id, name, email, password)

VALUES

(1, 'John Doe', 'john.doe@example.com', '123'),

(2, 'Jane Smith', 'jane.smith@example.com', '1234'),

(3, 'Alice Brown', 'alice.brown@example.com', '12345),

(4, 'Bob Johnson', 'bob.johnson@example.com', '123456'),

(5, 'Eve Davis', 'eve.davis@example.com', '12345677'),

(6, 'Charlie White', 'charlie.white@example.com', 'password'),

(7, 'Diana Clark', 'diana.clark@example.com', 'abc'),

(8, 'Frank Moore', 'frank.moore@example.com', 'bcd'),

(9, 'Grace Green', 'grace.green@example.com', '1234'),

(10, 'Henry King', 'henry.king@example.com', '12345');

INSERT INTO products (product\_id, name, price, description, stockQuantity)

VALUES

(1, 'Refrigerator', 800, 'A double door refrigerator', 5),

(2, 'Laptop', 1200, 'High performance laptop', 10),

(3, 'Smartphone', 700, 'Latest model smartphone', 20),

(4, 'Microwave Oven', 150, 'Compact microwave oven', 8),

(5, 'Washing Machine', 500, 'Front load washing machine', 4),

(6, 'Television', 900, 'LED TV with 4K display', 6),

(7, 'Air Conditioner', 1000, 'Split AC with fast cooling', 3),

(8, 'Vacuum Cleaner', 200, 'Handheld vacuum cleaner', 12),

(9, 'Blender', 80, 'High-speed kitchen blender', 15),

(10, 'Smartwatch', 300, 'Waterproof smartwatch', 25);

INSERT INTO cart (cart\_id, customer\_id, product\_id, quantity)

VALUES

(1, 1, 2, 1),

(2, 1, 3, 2),

(3, 2, 5, 1),

(4, 2, 1, 1),

(5, 3, 8, 3),

(6, 4, 7, 1),

(7, 5, 6, 2),

(8, 6, 4, 1),

(9, 7, 9, 2),

(10, 8, 10, 1);

INSERT INTO order\_items (order\_item\_id, order\_id, product\_id, quantity, amount)

VALUES

(1, 1, 2, 2, 2400.00),

(2, 1, 3, 1, 700.00),

(3, 2, 5, 1, 500.00),

(4, 2, 1, 1, 800.00),

(5, 3, 7, 1, 1000.00),

(6, 3, 9, 2, 160.00),

(7, 4, 6, 1, 900.00),

(8, 5, 4, 1, 150.00),

(9, 5, 10, 1, 300.00),

(10, 6, 8, 1, 200.00);

# Queries:

**1. Update refrigerator product price to 800.**

Ans- update products

set price=800

where name='refrigirator';

**2. Remove all cart items for a specific customer.**

Ans- DELETE FROM cart

WHERE customer\_id = 5;

**3. Retrieve Products Priced Below $100**

Ans- select \* from products

where price < 100;

.

**4. Find Products with Stock Quantity Greater Than 5.**

Ans- select \* from products

where stockQuantity > 5 ;

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

Ans- select \* from orders

where total\_price between 500 and 1000;

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

Ans- select \* from products

where name like '%r';

**7. Retrieve Cart Items for Customer 5.**

Ans- select \* from cart

where customer\_id=5;

**8. Find Customers Who Placed Orders in 2023.**

Ans- select \* from orders

where order\_date like '2023%';

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

Ans- select category, min(p.stockQuantity) as min\_stock from products p

group by category;

**10. Calculate the Total Amount Spent by Each Customer.**

Ans- select c.name, o.customer\_id, sum(o.total\_price) as money\_spent

from orders o

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

group by o.customer\_id;

**11. Find the Average Order Amount for Each Customer.**

Ans- SELECT o.customer\_id, c.name, AVG(o.total\_price) AS avg\_order\_amount

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY o.customer\_id;

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

Ans- SELECT o.customer\_id, c.name, COUNT(o.order\_id) AS count\_order

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY o.customer\_id;

**13. Find the Maximum Order Amount for Each Customer.**

select o.customer\_id, c.name, Max(total\_price) as highest\_order\_amount

from orders o

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

group by o.customer\_id;

**14. Get Customers Who Placed Orders Totaling Over $1000.**

Ans- select o.customer\_id, c.name, sum(o.total\_price) as spent

from orders o

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

group by o.customer\_id

having Sum(o.total\_price)>1000;

**15. Subquery to Find Products Not in the Cart.**

Ans- select \*

from products

where product\_id NOT IN (select product\_id from cart);

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

Ans- SELECT \*

FROM customers

WHERE customer\_id NOT IN (SELECT customer\_id FROM orders);

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

Ans- select p.product\_id, p.name,

(sum(oi.quantity\*p.price)/(select sum(total\_price) from orders))\*100 as percentage\_revenue

from products p

join order\_items oi on p.product\_id=oi.product\_id

group by p.product\_id,p.name;

**18. Subquery to Find Products with Low Stock.**

Ans- SELECT \*

FROM products

WHERE stockQuantity < (SELECT AVG(stockQuantity) FROM products);

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

Ans- select \*

from customers

where customer\_id IN (SELECT customer\_id from orders where total\_price > 1000);