Task3

Table Details with data :

```
CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(15),
    address TEXT
);
CREATE TABLE products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    description TEXT,
    price DECIMAL(10, 2),
    stock INT
);
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    status VARCHAR(20),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
CREATE TABLE order_items (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
```

```
    quantity INT,
    price DECIMAL(10, 2),
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
CREATE TABLE payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    payment_date DATE,
    amount DECIMAL(10, 2),
    method VARCHAR(50),
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
);

INSERT INTO customers (name, email, phone, address) VALUES
('Roberto Allen', 'andrewedwards@gutierrez.biz', '970.283.8578', '497 Susan Harbors Suite 344, Lake Janet, NJ 46219'),
('Taylor Griffith', 'berryamy@yahoo.com', '(240)415-7473', 'USNV Taylor, FPO AP 97611'),
('Dana Santiago', 'briangutierrez@kennedy.net', '(677)345-4019', '62067 Sharon Well, Nancytown, OK 79508'),
('Melissa Brewer', 'nmartinez@thomas.org', '958.784.1410', '8425 Roy Ridge Apt. 687, Michaelmouth, AL 08104'),
('Amanda George', 'nsmith@hotmail.com', '001-563-019-1801x791', '6302 Lambert Dale, Port Linda, CT 22455'),
('Cory Rose', 'robert07@yahoo.com', '920.282.9283', '83890 Mark Underpass, Reyesfort, OR 29192'),
('Jason Mitchell', 'gomezjohn@kennedy.org', '978.409.7433', '7319 Walter Fort Apt. 772, Kimberlyview, WY 52020'),
```

('Lisa Hawkins', 'kellyhernandez@gmail.com', '201-958-1559', '2545 Jackson Hill, East Steven, SD 36784'),

('Michael Brown', 'laura41@wade.net', '663-487-8720', '938 Dana View Suite 841, New Laura, NH 82166'),

('Brittany Price', 'nicholas95@bailey.org', '1-207-836-4527', '8989 Elizabeth Rapids, Cruzport, IL 12057');

INSERT INTO products (name, description, price, stock) VALUES

('Laptop', 'High-performance laptop with 16GB RAM.', 899.99, 30),

('Smartphone', 'Latest smartphone with 128GB storage.', 699.00, 50),

('Tablet', 'Portable tablet with retina display.', 349.49, 40),

('Monitor', '24-inch Full HD LED monitor.', 199.99, 25),

('Keyboard', 'Mechanical RGB keyboard.', 89.95, 100),

('Mouse', 'Wireless ergonomic mouse.', 39.90, 80),

('Headphones', 'Noise-cancelling headphones.', 129.99, 60),

('Speaker', 'Bluetooth speaker with bass boost.', 49.99, 70),

('Smartwatch', 'Fitness tracking smartwatch.', 159.95, 35),

('Printer', 'All-in-one inkjet printer.', 119.89, 20);

INSERT INTO orders (order_id, customer_id, order_date, status) VALUES

(1, 2, '2024-01-17', 'Cancelled'),

(2, 5, '2023-10-23', 'Delivered'),

(3, 1, '2023-06-24', 'Shipped'),

(4, 4, '2023-12-12', 'Pending'),

(5, 8, '2024-05-05', 'Delivered'),

(6, 3, '2024-02-18', 'Cancelled'),

(7, 7, '2023-11-30', 'Shipped'),

(8, 9, '2023-09-10', 'Pending'),

(9, 6, '2024-04-22', 'Delivered'),

(10, 10, '2023-08-05', 'Delivered');

```
INSERT INTO order_items (item_id, order_id, product_id, quantity, price) VALUES
(1, 1, 3, 2, 349.49),
(2, 2, 6, 1, 39.90),
(3, 3, 1, 1, 899.99),
(4, 4, 5, 3, 89.95),
(5, 5, 2, 2, 699.00),
(6, 6, 8, 1, 49.99),
(7, 7, 9, 1, 159.95),
(8, 8, 7, 2, 129.99),
(9, 9, 4, 1, 199.99),
(10, 10, 10, 1, 119.89);

INSERT INTO payments (payment_id, order_id, payment_date, amount, method)
VALUES
(1, 1, '2024-01-20', 698.98, 'Credit Card'),
(2, 2, '2023-10-25', 39.90, 'UPI'),
(3, 3, '2023-06-27', 899.99, 'Net Banking'),
(4, 4, '2023-12-13', 269.85, 'Credit Card'),
(5, 5, '2024-05-07', 1398.00, 'Debit Card'),
(6, 6, '2024-02-20', 49.99, 'Cash on Delivery'),
(7, 7, '2023-12-01', 159.95, 'Credit Card'),
(8, 8, '2023-09-12', 259.98, 'UPI'),
(9, 9, '2024-04-24', 199.99, 'Net Banking'),
(10, 10, '2023-08-06', 119.89, 'Credit Card');
```

Querys I have used:

```
-- Get all customers from 'New York'
SELECT * FROM customers
WHERE address LIKE '%NY%';

-- List orders sorted by most recent
SELECT * FROM orders
ORDER BY order_date DESC;

-- Total number of orders by status
SELECT status, COUNT(*) AS total_orders
FROM orders
GROUP BY status;

-- INNER JOIN: Orders with customer names
SELECT o.order_id, c.name, o.order_date, o.status
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id;

-- LEFT JOIN: All customers and their orders (if any)
SELECT c.name, o.order_id, o.status
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;

-- RIGHT JOIN: All orders and their customers (in case customers table is incomplete)
SELECT o.order_id, c.name
FROM customers c
RIGHT JOIN orders o ON c.customer_id = o.customer_id;
```

```sql
-- Customers who placed more than 2 orders
SELECT name FROM customers
WHERE customer_id IN (
   SELECT customer_id
   FROM orders
   GROUP BY customer_id
   HAVING COUNT(order_id) > 2
);

-- Orders with total value greater than 500
SELECT o.order_id, o.customer_id
FROM orders o
WHERE (
   SELECT SUM(price * quantity)
   FROM order_items
   WHERE order_id = o.order_id
) > 500;

-- Total sales amount
SELECT SUM(amount) AS total_sales FROM payments;

-- Average order amount per customer
SELECT customer_id, AVG(amount) AS avg_order_value
FROM payments p
JOIN orders o ON p.order_id = o.order_id
GROUP BY customer_id;

-- Total quantity sold per product
```

```
SELECT p.name, SUM(oi.quantity) AS total_sold
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.name;

-- View for customer order history
CREATE VIEW customer_orders AS
SELECT c.name, o.order_id, o.order_date, o.status
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id;

-- View for product sales
CREATE VIEW product_sales AS
SELECT p.product_id, p.name, SUM(oi.quantity) AS units_sold, SUM(oi.price *
oi.quantity) AS revenue
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.name;

-- Index on customer_id for faster JOINs and WHEREs
CREATE INDEX idx_customer_id ON customers(customer_id);

-- Index on order_date for faster sorting/filtering
CREATE INDEX idx_order_date ON orders(order_date);

-- Index on product_id in order_items
CREATE INDEX idx_product_id ON order_items(product_id);
```