Database Schema

1. Customers Table:
   - `customer_id` (Primary Key)
   - `name`
   - `email`

2. Products Table:
   - `product_id` (Primary Key)
   - `product_name`
   - `price`

3. Orders Table:
   - `order_id` (Primary Key)
   - `customer_id` (Foreign Key referencing `customers` table)
   - `product_id` (Foreign Key referencing `products` table)
   - `quantity`
   - `order_date`

SQL Queries

1. Insert Data:

```sql
-- Insert data into customers table
INSERT INTO customers (name, email) VALUES ('John Doe', 'john@example.com');
INSERT INTO customers (name, email) VALUES ('Jane Smith', 'jane@example.com');

-- Insert data into products table
INSERT INTO products (product_name, price) VALUES ('Laptop', 999.99);
INSERT INTO products (product_name, price) VALUES ('Smartphone', 499.99);

-- Insert data into orders table
INSERT INTO orders (customer_id, product_id, quantity, order_date) VALUES (1, 1, 2, '2024-06-01');
```

```
INSERT INTO orders (customer_id, product_id, quantity, order_date) VALUES (2, 2, 1, '2024-06-02');
```

2. Basic Retrieval:

```sql
-- Retrieve all orders
SELECT * FROM orders;

-- Retrieve orders placed by a specific customer
SELECT * FROM orders WHERE customer_id = 1;

-- Retrieve orders for a specific product
SELECT * FROM orders WHERE product_id = 2;
```

3. Filtering and Sorting:

```sql
-- Retrieve orders placed after a specific date
SELECT * FROM orders WHERE order_date > '2024-06-01';

-- Retrieve orders with a quantity greater than a certain value
SELECT * FROM orders WHERE quantity > 1;

-- Retrieve orders sorted by order date in descending order
SELECT * FROM orders ORDER BY order_date DESC;
```

4. Joins:

```sql
-- Retrieve orders along with customer details
SELECT orders.*, customers.name AS customer_name, customers.email
```

```
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;

-- Retrieve orders along with product details
SELECT orders.*, products.product_name, products.price
FROM orders
INNER JOIN products ON orders.product_id = products.product_id;
```

5. Aggregation and Subqueries:

```sql
-- Calculate the total revenue generated by each customer
SELECT customers.name, SUM(products.price * orders.quantity) AS total_revenue
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN products ON orders.product_id = products.product_id
GROUP BY customers.name;

-- Retrieve customers who have placed orders
SELECT * FROM customers WHERE customer_id IN (SELECT DISTINCT customer_id FROM orders);

-- Retrieve products that have been ordered more than 10 times
SELECT * FROM products WHERE product_id IN (SELECT product_id FROM orders GROUP BY product_id HAVING COUNT(*) > 10);
```

6. Data Modification and Advanced Retrieval:

```sql
-- Update the quantity of a specific order
UPDATE orders SET quantity = 3 WHERE order_id = 1;
```

```
-- Delete an order from the database
DELETE FROM orders WHERE order_id = 2;

-- Retrieve orders along with customer names and product names
SELECT orders.order_id, customers.name AS customer_name, products.product_name, products.price, orders.quantity
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN products ON orders.product_id = products.product_id;

-- Retrieve orders where the total order amount is greater than $100
SELECT orders.*, customers.name AS customer_name, products.product_name, products.price, orders.quantity
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id
INNER JOIN products ON orders.product_id = products.product_id
WHERE (products.price * orders.quantity) > 100;

-- Retrieve orders placed within a specific date range
SELECT * FROM orders WHERE order_date BETWEEN '2024-06-01' AND '2024-06-30';
```