A. Research

1. Difference between delete and truncate?

   DELETE removes specific rows from a table, supports a WHERE clause, logs each deletion, and can be rolled back within a transaction. It activates triggers and can be slower for large datasets.

2. Execution order for query processing?

   1. **FROM**: Specifies the tables involved and performs any joins to produce the base data set.
   2. **WHERE**: Filters the rows of the base data set that meet certain conditions.
   3. **GROUP BY**: Groups the rows based on the values of specified columns.
   4. **HAVING**: Filters the groups created by the `GROUP BY` clause based on specified conditions.
   5. **SELECT**: Selects the columns or expressions to be returned in the result set.
   6. **DISTINCT**: Removes duplicate rows from the result set.
   7. **ORDER BY**: Sorts the result set based on specified columns or expressions.
   8. **LIMIT** / **OFFSET**: Limits the number of rows returned and optionally skips a specified number of rows.

3. Difference between union and union all ?

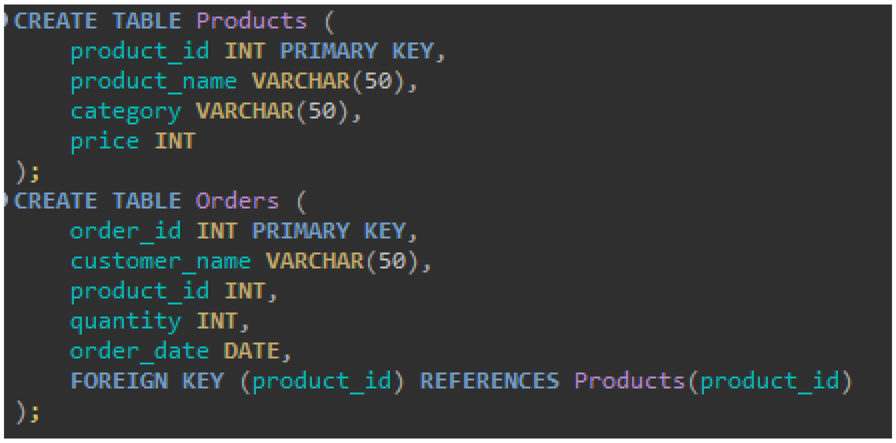
   UNION: Combines results and removes duplicates, potentially slower.

   UNION ALL: Combines results and retains duplicates, generally faster.

B. Create two tables

Products: columns (product_id,product_name,category and price)

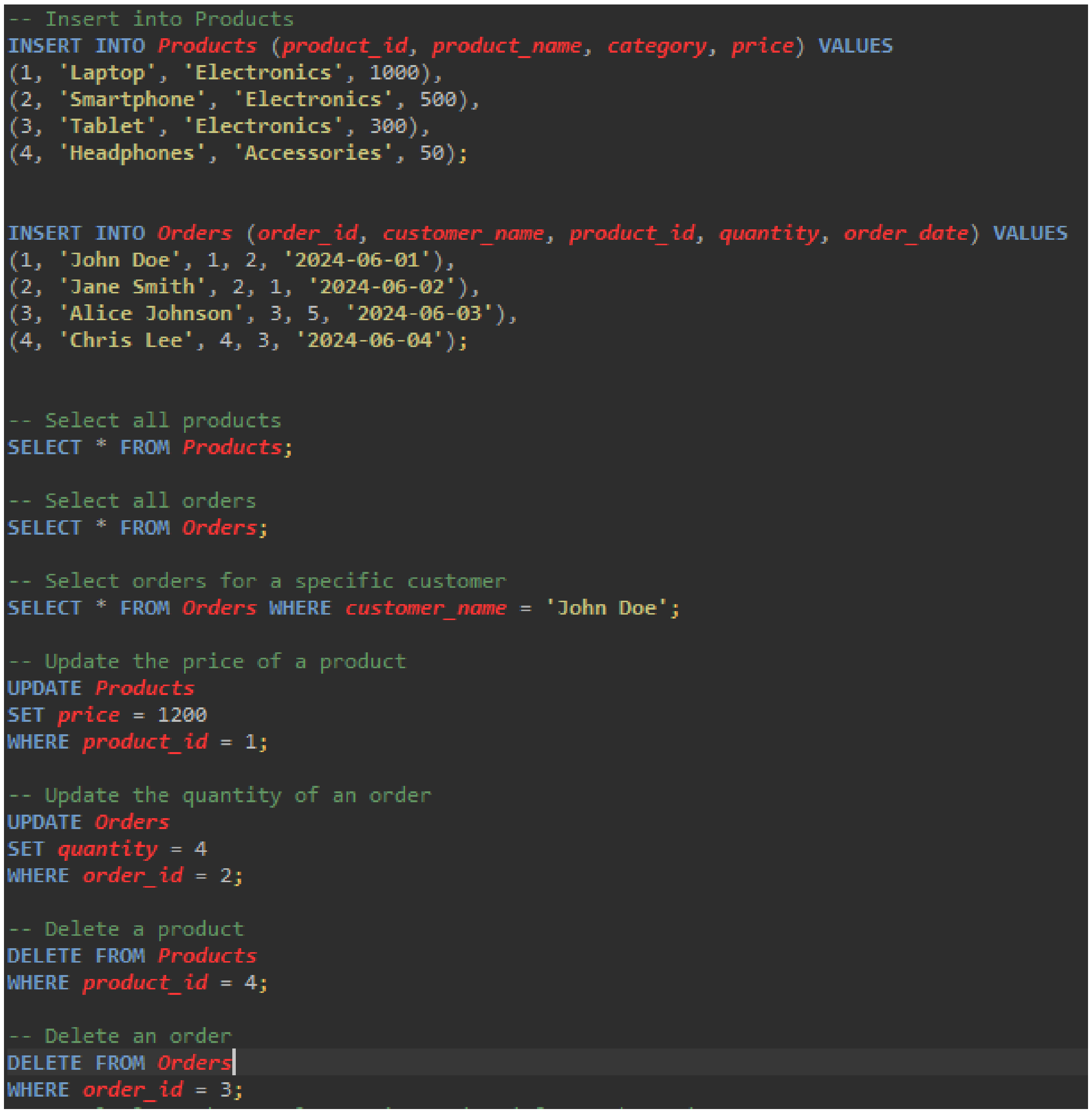
Orders : columns(order_id, customer_name, product_id, quantity, order_date)



```sql
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(50),
    category VARCHAR(50),
    price INT
);
CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_name VARCHAR(50),
    product_id INT,
    quantity INT,
    order_date DATE,
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

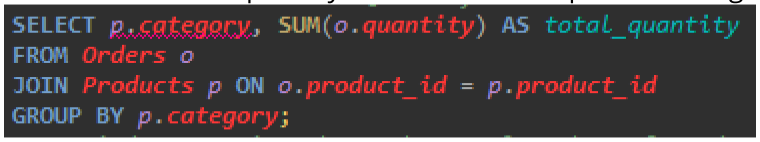
C. QUESTIONS:

1. perform CRUD

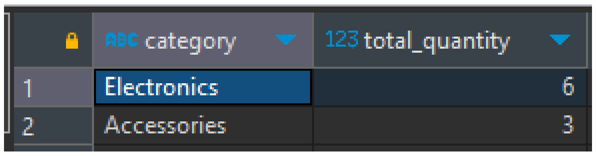


```sql
-- Insert into Products
INSERT INTO Products (product_id, product_name, category, price) VALUES
(1, 'Laptop', 'Electronics', 1000),
(2, 'Smartphone', 'Electronics', 500),
(3, 'Tablet', 'Electronics', 300),
(4, 'Headphones', 'Accessories', 50);


INSERT INTO Orders (order_id, customer_name, product_id, quantity, order_date) VALUES
(1, 'John Doe', 1, 2, '2024-06-01'),
(2, 'Jane Smith', 2, 1, '2024-06-02'),
(3, 'Alice Johnson', 3, 5, '2024-06-03'),
(4, 'Chris Lee', 4, 3, '2024-06-04');


-- Select all products
SELECT * FROM Products;

-- Select all orders
SELECT * FROM Orders;

-- Select orders for a specific customer
SELECT * FROM Orders WHERE customer_name = 'John Doe';

-- Update the price of a product
UPDATE Products
SET price = 1200
WHERE product_id = 1;

-- Update the quantity of an order
UPDATE Orders
SET quantity = 4
WHERE order_id = 2;

-- Delete a product
DELETE FROM Products
WHERE product_id = 4;

-- Delete an order
DELETE FROM Orders
WHERE order_id = 3;
```

2. Calculate the total quantity ordered for each product category in the orders table.

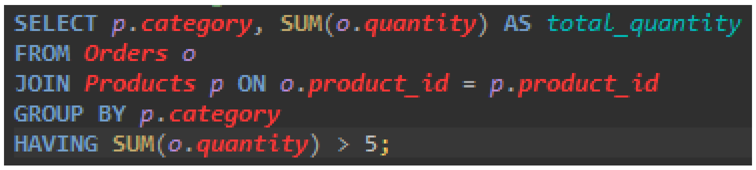


```
SELECT p.category, SUM(o.quantity) AS total_quantity
FROM Orders o
JOIN Products p ON o.product_id = p.product_id
GROUP BY p.category;
```

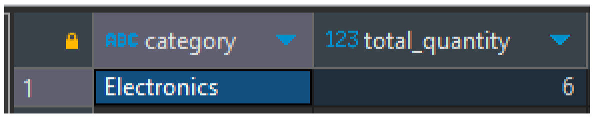


	category	total_quantity
1	Electronics	6
2	Accessories	3

3. Find categories where the total number of products ordered is greater than 5.



```
SELECT p.category, SUM(o.quantity) AS total_quantity
FROM Orders o
JOIN Products p ON o.product_id = p.product_id
GROUP BY p.category
HAVING SUM(o.quantity) > 5;
```



	category	total_quantity
1	Electronics	6