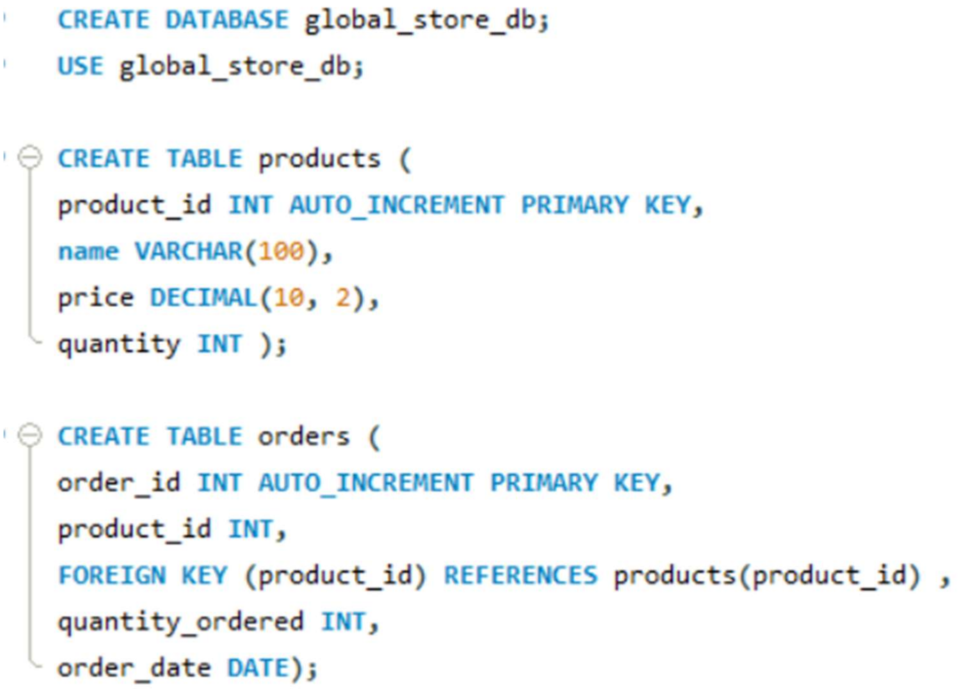
# MINI PROJECT

**1. Create the following tables inside the database ‘global_store_db’.**
**‘products’ with columns:**

- **product_id (INT, auto_increment, primary key),**
- **name (VARCHAR(100)),**
- **price (DECIMAL(10,2)),**
- **quantity (INT).**

**‘orders’ with columns:**

- **order_id (INT, auto_increment, primary key),**
- **product_id (INT, foreign key referencing product_id in the inventory table),**
- **quantity_ordered (INT)**
- **order_date (DATE).**

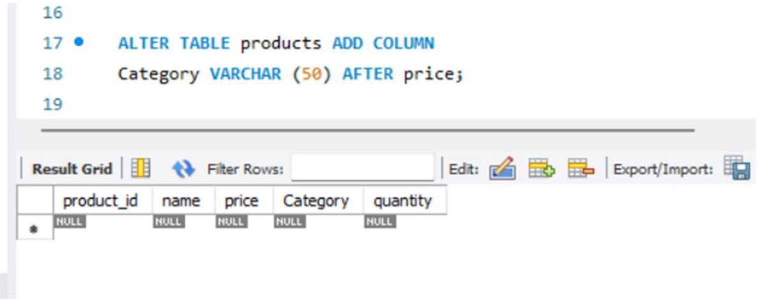


```
CREATE DATABASE global_store_db;
USE global_store_db;

CREATE TABLE products (
product_id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100),
price DECIMAL(10, 2),
quantity INT );

CREATE TABLE orders (
order_id INT AUTO_INCREMENT PRIMARY KEY,
product_id INT,
FOREIGN KEY (product_id) REFERENCES products(product_id) ,
quantity_ordered INT,
order_date DATE);
```

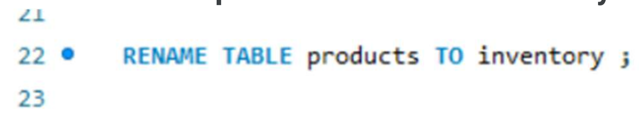
**2.Alter the products table to add a new column named category (VARCHAR(50)) after the price column.**



```sql
ALTER TABLE products ADD COLUMN
Category VARCHAR (50) AFTER price;
```

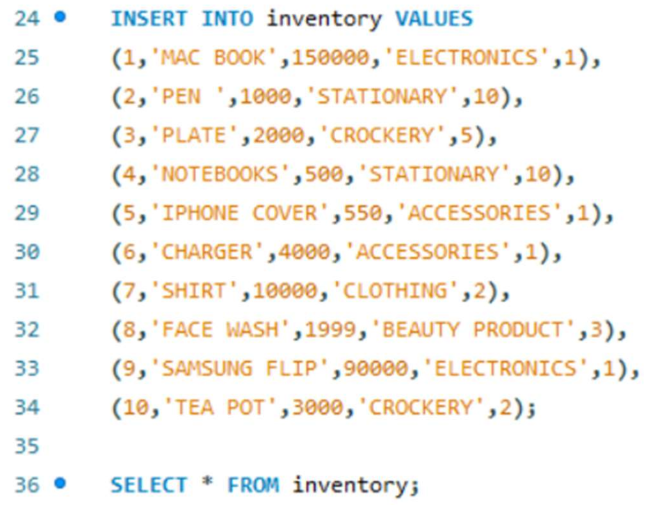
| product_id | name | price | Category | quantity |
|---|---|---|---|---|
| NULL | NULL | NULL | NULL | NULL |

### 3. Rename the products table to inventory.



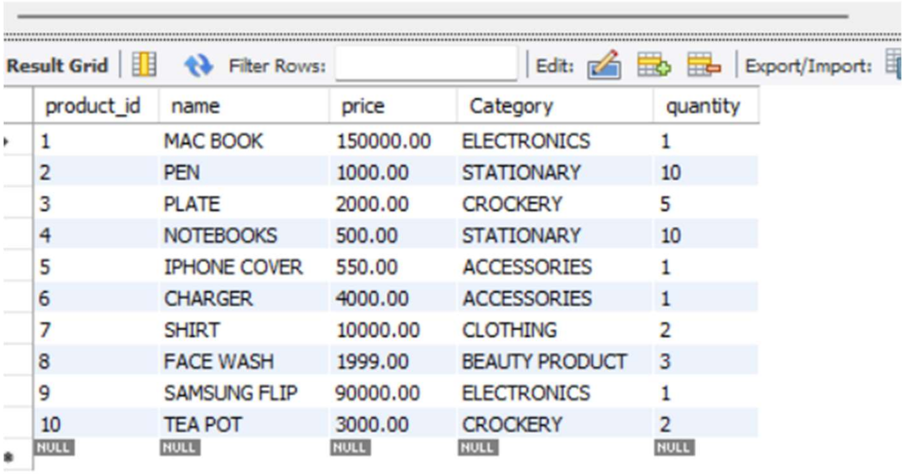
```sql
RENAME TABLE products TO inventory ;
```

### 4. Insert at least 10 records into the inventory table and 5 records into orders table and display the tables.

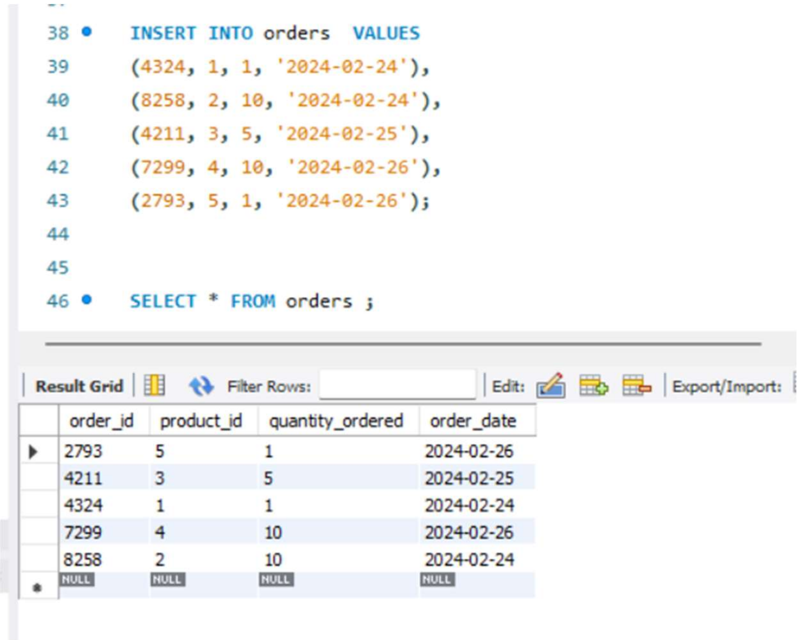


```sql
INSERT INTO inventory VALUES
(1,'MAC BOOK',150000,'ELECTRONICS',1),
(2,'PEN ',1000,'STATIONARY',10),
(3,'PLATE',2000,'CROCKERY',5),
(4,'NOTEBOOKS',500,'STATIONARY',10),
(5,'IPHONE COVER',550,'ACCESSORIES',1),
(6,'CHARGER',4000,'ACCESSORIES',1),
(7,'SHIRT',10000,'CLOTHING',2),
(8,'FACE WASH',1999,'BEAUTY PRODUCT',3),
(9,'SAMSUNG FLIP',90000,'ELECTRONICS',1),
(10,'TEA POT',3000,'CROCKERY',2);

SELECT * FROM inventory;
```



| product_id | name | price | Category | quantity |
|---|---|---|---|---|
| 1 | MAC BOOK | 150000.00 | ELECTRONICS | 1 |
| 2 | PEN | 1000.00 | STATIONARY | 10 |
| 3 | PLATE | 2000.00 | CROCKERY | 5 |
| 4 | NOTEBOOKS | 500.00 | STATIONARY | 10 |
| 5 | IPHONE COVER | 550.00 | ACCESSORIES | 1 |
| 6 | CHARGER | 4000.00 | ACCESSORIES | 1 |
| 7 | SHIRT | 10000.00 | CLOTHING | 2 |
| 8 | FACE WASH | 1999.00 | BEAUTY PRODUCT | 3 |
| 9 | SAMSUNG FLIP | 90000.00 | ELECTRONICS | 1 |
| 10 | TEA POT | 3000.00 | CROCKERY | 2 |
| NULL | NULL | NULL | NULL | NULL |



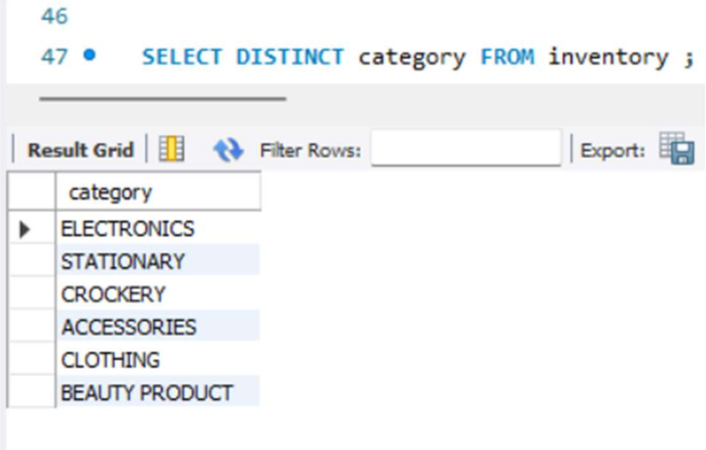
```sql
INSERT INTO orders  VALUES
(4324, 1, 1, '2024-02-24'),
(8258, 2, 10, '2024-02-24'),
(4211, 3, 5, '2024-02-25'),
(7299, 4, 10, '2024-02-26'),
(2793, 5, 1, '2024-02-26');


SELECT * FROM orders ;
```

| order_id | product_id | quantity_ordered | order_date |
|---|---|---|---|
| 2793 | 5 | 1 | 2024-02-26 |
| 4211 | 3 | 5 | 2024-02-25 |
| 4324 | 1 | 1 | 2024-02-24 |
| 7299 | 4 | 10 | 2024-02-26 |
| 8258 | 2 | 10 | 2024-02-24 |
| NULL | NULL | NULL | NULL |

**5. Write queries for the following :**

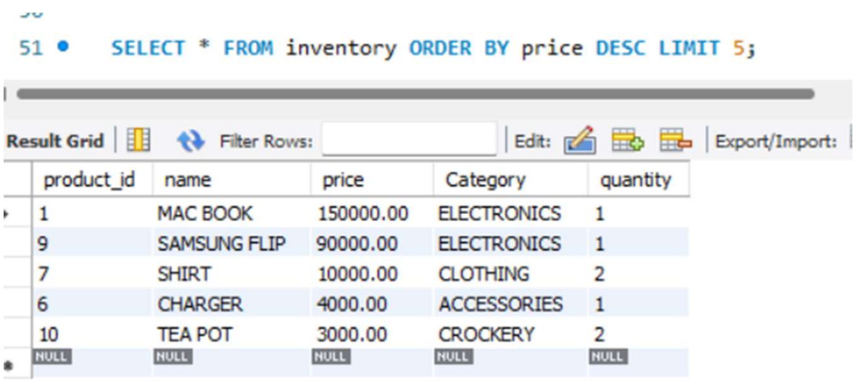
**a) Write a query to display distinct categories from the inventory table.**



```sql
SELECT DISTINCT category FROM inventory ;
```

| category |
|---|
| ELECTRONICS |
| STATIONARY |
| CROCKERY |
| ACCESSORIES |
| CLOTHING |
| BEAUTY PRODUCT |

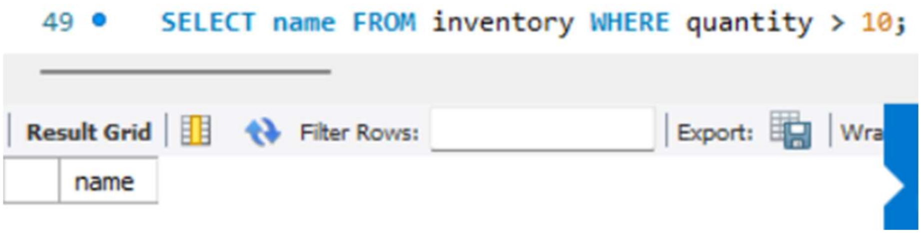
**b) Select the top 5 products by their prices in descending order from the inventory table.**



```sql
SELECT * FROM inventory ORDER BY price DESC LIMIT 5;
```

| product_id | name | price | Category | quantity |
|---|---|---|---|---|
| 1 | MAC BOOK | 150000.00 | ELECTRONICS | 1 |
| 9 | SAMSUNG FLIP | 90000.00 | ELECTRONICS | 1 |
| 7 | SHIRT | 10000.00 | CLOTHING | 2 |
| 6 | CHARGER | 4000.00 | ACCESSORIES | 1 |
| 10 | TEA POT | 3000.00 | CROCKERY | 2 |
| NULL | NULL | NULL | NULL | NULL |

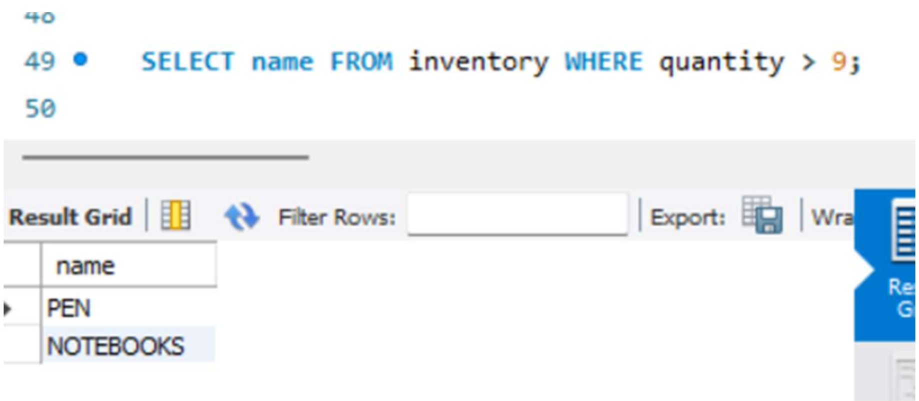
c) Display the names of products with a quantity greater than 10 from the inventory table.



```sql
SELECT name FROM inventory WHERE quantity > 10;
```

| name |
|---|

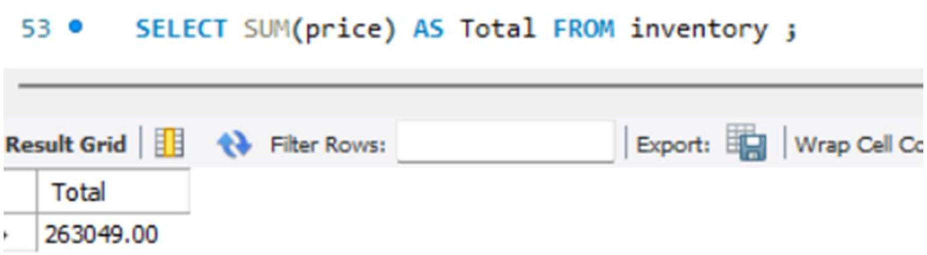
!! OR WHICH IS GREATER THAN 9 (only for verification)



```sql
SELECT name FROM inventory WHERE quantity > 9;
```

| name |
|---|
| PEN |
| NOTEBOOKS |

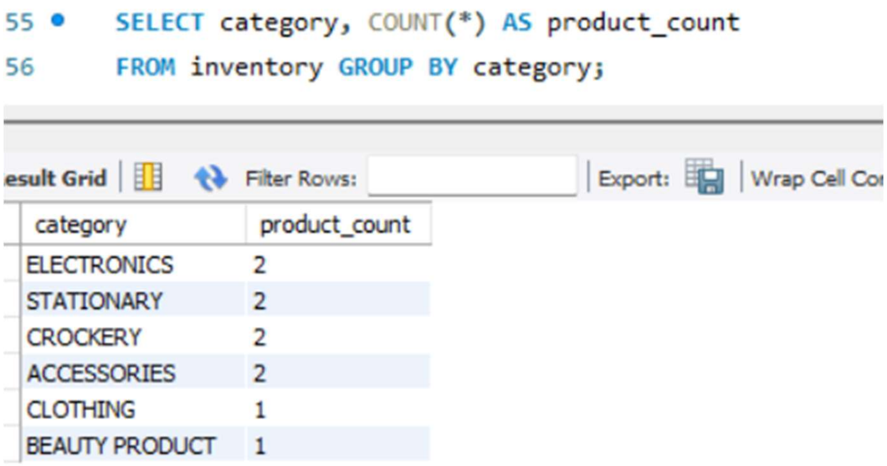
d) Use the SUM() function to calculate the total price of all products in the inventory table.



```sql
SELECT SUM(price) AS Total FROM inventory ;
```

| Total |
|---|
| 263049.00 |

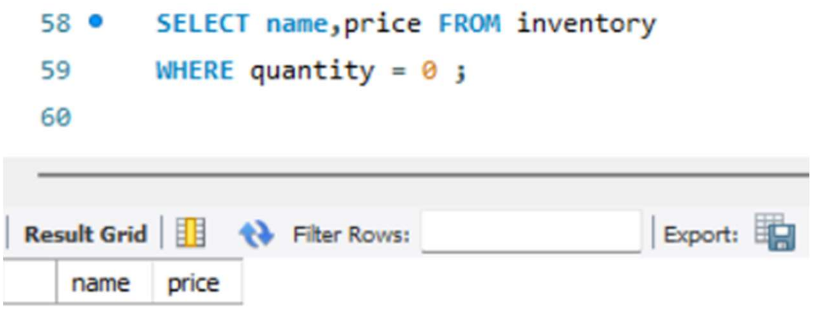
e )Group products by their categories and display the count of products in each category.



```sql
SELECT category, COUNT(*) AS product_count
FROM inventory GROUP BY category;
```

| category | product_count |
|---|---|
| ELECTRONICS | 2 |
| STATIONARY | 2 |
| CROCKERY | 2 |
| ACCESSORIES | 2 |
| CLOTHING | 1 |
| BEAUTY PRODUCT | 1 |

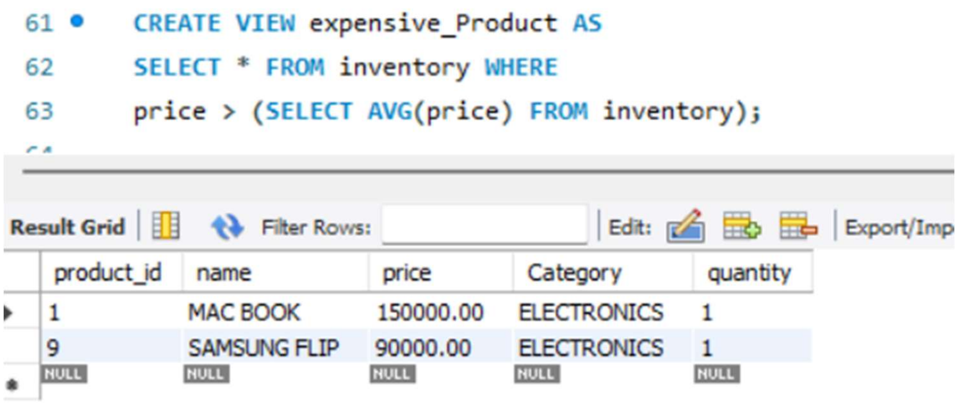
f) Write a query to identify products that are currently out of stock (i.e., quantity is zero). Display the product details including the product name and price.



```sql
SELECT name,price FROM inventory
WHERE quantity = 0 ;
```

| name | price |
|---|---|

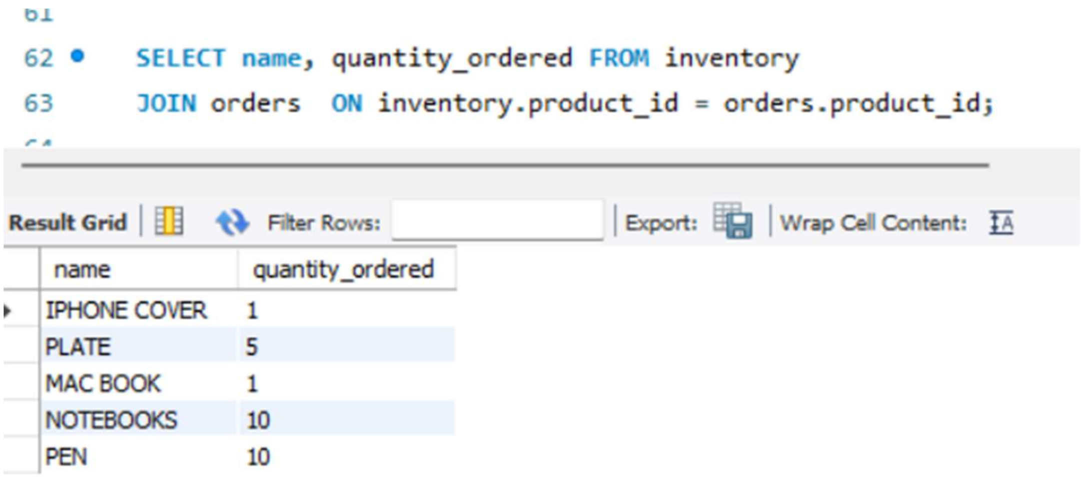
6. Create a view named expensive_products that displays the details of products with a price above the average price of all products.



```sql
CREATE VIEW expensive_Product AS
SELECT * FROM inventory WHERE
price > (SELECT AVG(price) FROM inventory);
```

| product_id | name | price | Category | quantity |
|---|---|---|---|---|
| 1 | MAC BOOK | 150000.00 | ELECTRONICS | 1 |
| 9 | SAMSUNG FLIP | 90000.00 | ELECTRONICS | 1 |
| NULL | NULL | NULL | NULL | NULL |

**7. Write a join query to display the names of products along with the corresponding order quantities from the inventory and orders tables.**



```sql
SELECT name, quantity_ordered FROM inventory
JOIN orders  ON inventory.product_id = orders.product_id;
```

| name | quantity_ordered |
|---|---|
| IPHONE COVER | 1 |
| PLATE | 5 |
| MAC BOOK | 1 |
| NOTEBOOKS | 10 |
| PEN | 10 |