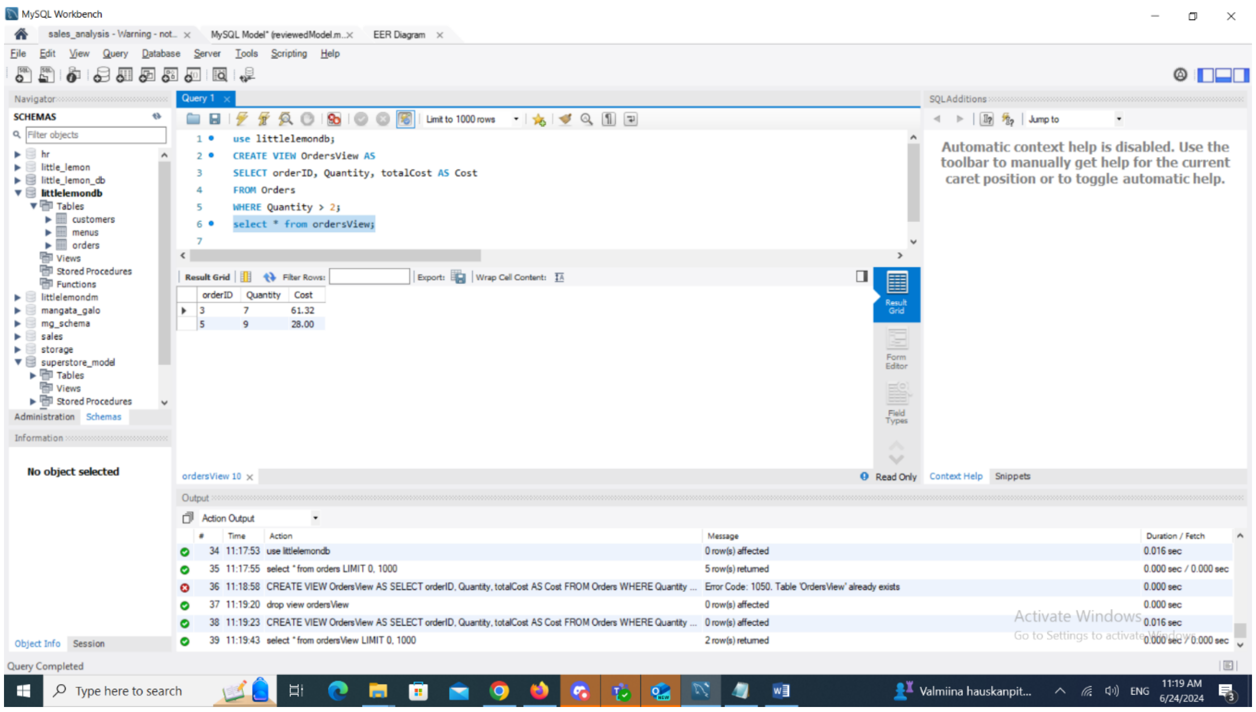
CREATE A VIRTUAL TABLE TO SUMMARIZE DATA:

Creating a virtual table called “OrdersView” that focuses on “OrderID”, Quantity and Cost columns within the Orders table for all orders with a quantity greater than 2.

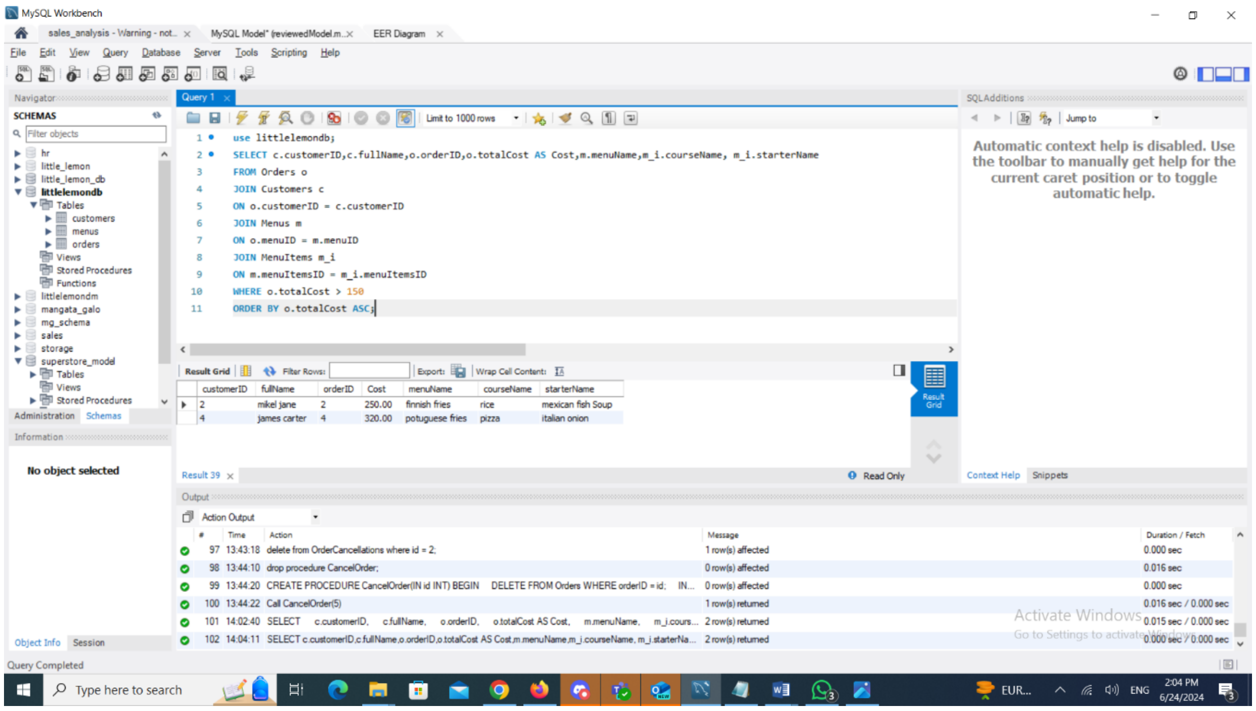


```sql
use littlelemondb;
CREATE VIEW OrdersView AS
SELECT orderID, Quantity, totalCost AS Cost
FROM Orders
WHERE Quantity > 2;
select * from ordersView;
```

orderID	Quantity	Cost
3	7	61.32
5	9	28.00

```
#   Time      Action                                                                               Message                                              Duration / Fetch
34  11:17:53  use littlelemondb                                                                    0 row(s) affected                                    0.016 sec
35  11:17:55  select * from orders LIMIT 0, 1000                                                   5 row(s) returned                                    0.000 sec / 0.000 sec
36  11:18:58  CREATE VIEW OrdersView AS SELECT orderID, Quantity, totalCost AS Cost FROM Orders WHERE Quantity ...  Error Code: 1050. Table 'OrdersView' already exists  0.000 sec
37  11:19:20  drop view ordersView                                                                 0 row(s) affected                                    0.000 sec
38  11:19:23  CREATE VIEW OrdersView AS SELECT orderID, Quantity, totalCost AS Cost FROM Orders WHERE Quantity ...  0 row(s) affected                                    0.016 sec
39  11:19:43  select * from ordersView LIMIT 0, 1000                                               2 row(s) returned                                    0.000 sec / 0.000 sec
```

Summary from all four tables on all customers with orders that cost more than $150.

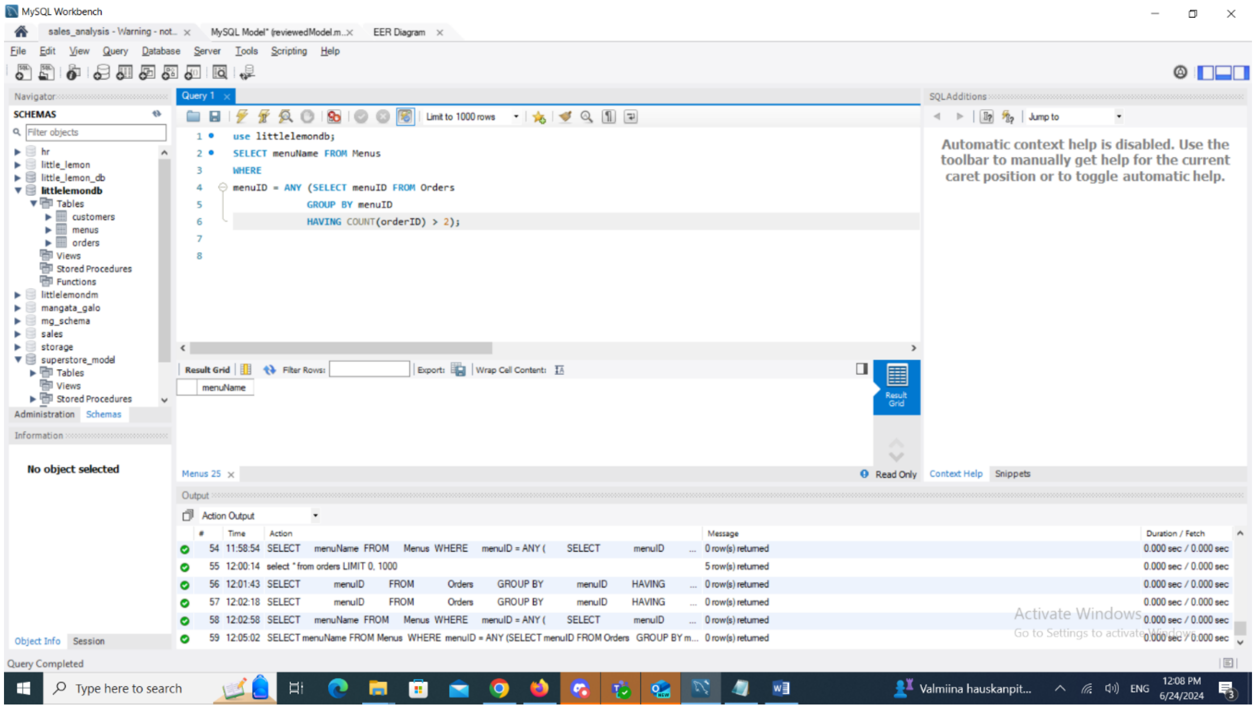


```sql
use littlelemondb;
SELECT c.customerID,c.fullName,o.orderID,o.totalCost AS Cost,m.menuName,m_i.courseName, m_i.starterName
FROM Orders o
JOIN Customers c
ON o.customerID = c.customerID
JOIN Menus m
ON o.menuID = m.menuID
JOIN MenuItems m_i
ON m.menuItemsID = m_i.menuItemsID
WHERE o.totalCost > 150
ORDER BY o.totalCost ASC;
```

customerID	fullName	orderID	Cost	menuName	courseName	starterName
2	mikel jane	2	250.00	finnish fries	rice	mexican fish Soup
4	james carter	4	320.00	potuguese fries	pizza	italian onion

```
#    Time      Action                                                                          Message             Duration / Fetch
97   13:43:18  delete from OrderCancellations where id = 2;                                    1 row(s) affected   0.000 sec
98   13:44:10  drop procedure CancelOrder;                                                     0 row(s) affected   0.016 sec
99   13:44:20  CREATE PROCEDURE CancelOrder(IN id INT) BEGIN    DELETE FROM Orders WHERE orderID = id;   IN...   0 row(s) affected   0.000 sec
100  13:44:22  Call CancelOrder(5)                                                             1 row(s) returned   0.016 sec / 0.000 sec
101  14:02:40  SELECT    c.customerID,    c.fullName,    o.orderID,    o.totalCost AS Cost,    m.menuName,    m_i.cours...   2 row(s) returned   0.015 sec / 0.000 sec
102  14:04:11  SELECT c.customerID,c.fullName,o.orderID,o.totalCost AS Cost,m.menuName,m_i.courseName, m_i.starterNa...   2 row(s) returned   0.000 sec / 0.000 sec
```

Find all menu items for which more than 2 orders have been placed.



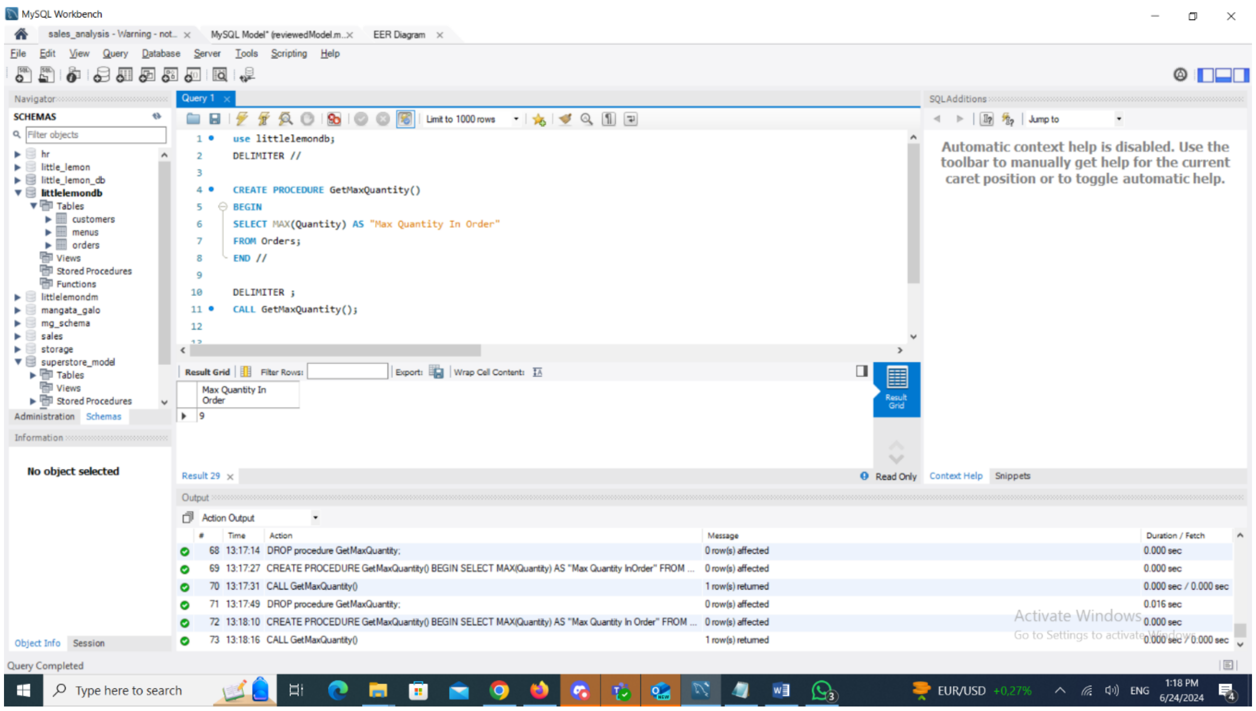
```sql
use littlelemondb;
SELECT menuName FROM Menus
WHERE
menuID = ANY (SELECT menuID FROM Orders
            GROUP BY menuID
            HAVING COUNT(orderID) > 2);
```

```
   #  Time      Action                                                                      Message            Duration / Fetch
   54 11:58:54  SELECT   menuName  FROM   Menus  WHERE   menuID = ANY (   SELECT   menuID  ...  0 row(s) returned  0.000 sec / 0.000 sec
   55 12:00:14  select * from orders LIMIT 0, 1000                                               5 row(s) returned  0.000 sec / 0.000 sec
   56 12:01:43  SELECT   menuID   FROM   Orders   GROUP BY   menuID   HAVING   ...               0 row(s) returned  0.000 sec / 0.000 sec
   57 12:02:18  SELECT   menuID   FROM   Orders   GROUP BY   menuID   HAVING   ...               0 row(s) returned  0.000 sec / 0.000 sec
   58 12:02:58  SELECT   menuName  FROM   Menus  WHERE   menuID = ANY (   SELECT   menuID  ...  0 row(s) returned  0.000 sec / 0.000 sec
   59 12:05:02  SELECT menuName FROM Menus WHERE menuID = ANY (SELECT menuID FROM Orders GROUP BY m...  0 row(s) returned  0.000 sec / 0.000 sec
```

MYSQL QUERY OPTIMIZATION WITH STORED PROCEDURES AND PREPARED STATEMENTS:

CREATING OPTIMIZED QUERIES TO MANAGE AND ANALYZE DATA:

Create a procedure that displays the maximum ordered quantity in the Orders table.

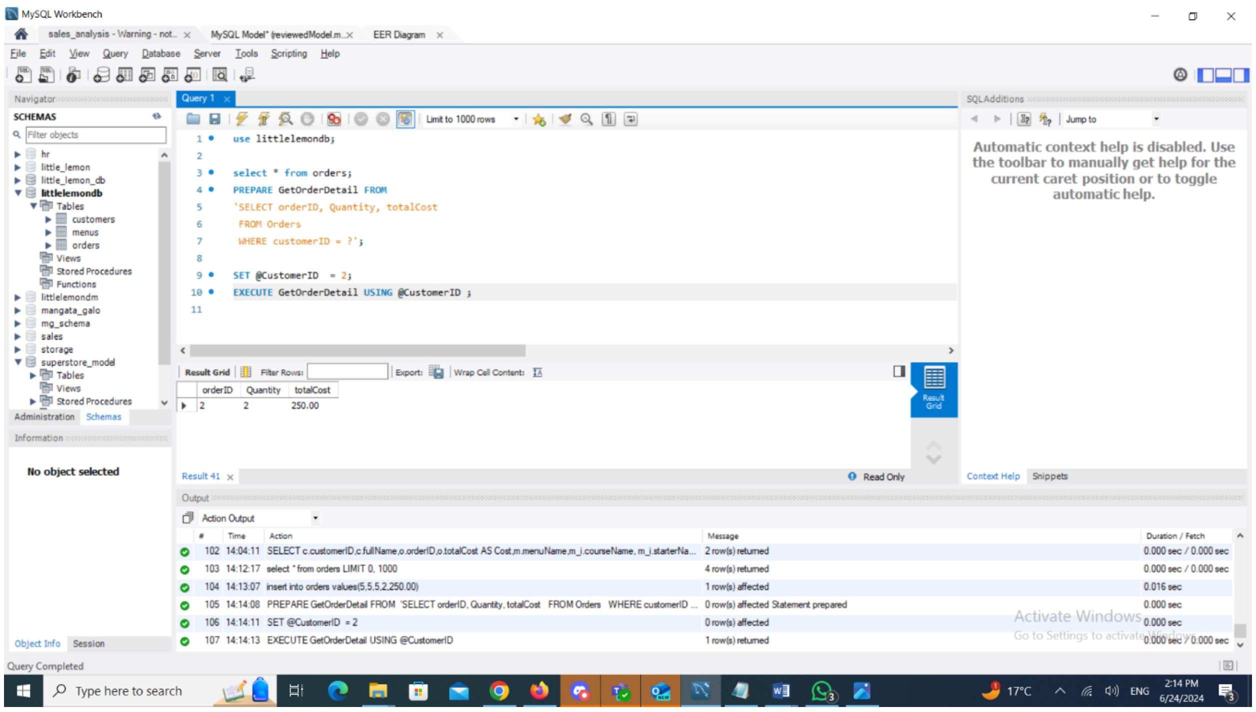


```
use littlelemondb;
DELIMITER //

CREATE PROCEDURE GetMaxQuantity()
BEGIN
SELECT MAX(Quantity) AS "Max Quantity In Order"
FROM Orders;
END //

DELIMITER ;
CALL GetMaxQuantity();
```

```
Max Quantity In Order
9
```

```
#   Time      Action                                                                                      Message              Duration / Fetch
68  13:17:14  DROP procedure GetMaxQuantity;                                                              0 row(s) affected    0.000 sec
69  13:17:27  CREATE PROCEDURE GetMaxQuantity() BEGIN SELECT MAX(Quantity) AS "Max Quantity InOrder" FROM ...   0 row(s) affected    0.000 sec
70  13:17:31  CALL GetMaxQuantity()                                                                       1 row(s) returned    0.000 sec / 0.000 sec
71  13:17:49  DROP procedure GetMaxQuantity;                                                              0 row(s) affected    0.016 sec
72  13:18:10  CREATE PROCEDURE GetMaxQuantity() BEGIN SELECT MAX(Quantity) AS "Max Quantity In Order" FROM ...  0 row(s) affected    0.000 sec
73  13:18:16  CALL GetMaxQuantity()                                                                       1 row(s) returned    0.000 sec / 0.000 sec
```

Create a prepared statement called *GetOrderDetail*. This prepared statement will help to reduce the parsing time of queries. It will also help to secure the database from SQL injections.

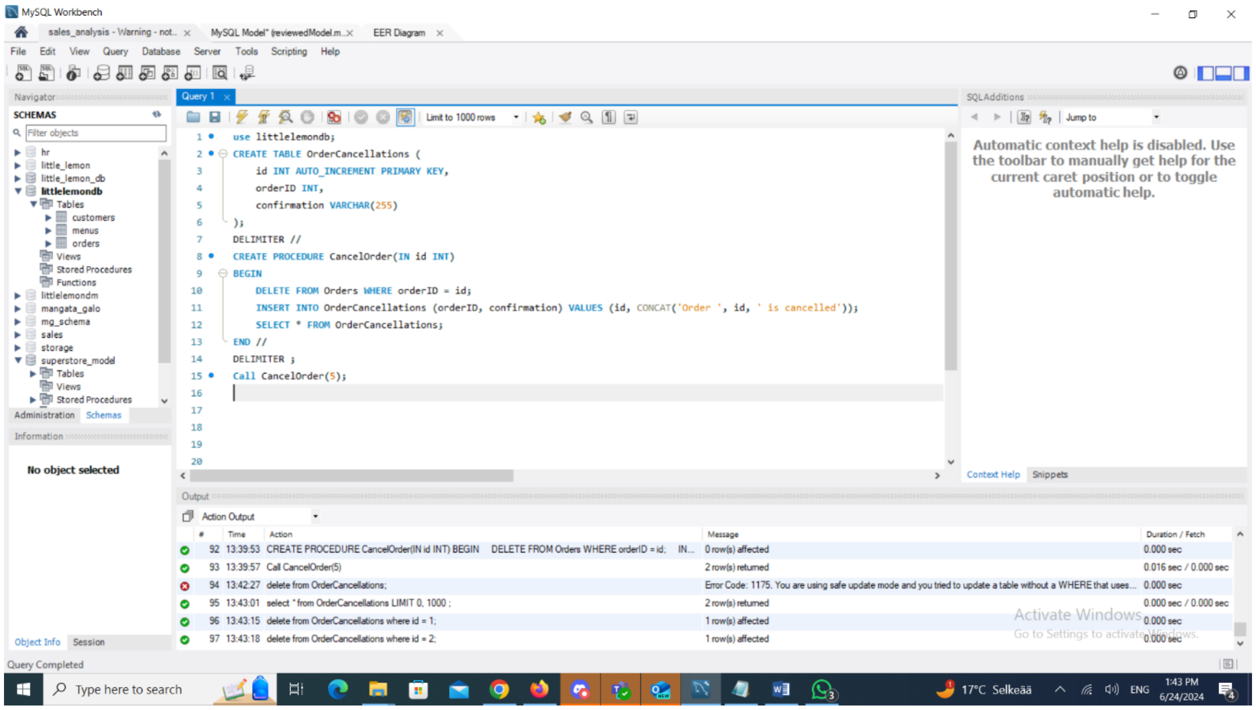


```
use littlelemondb;

select * from orders;
PREPARE GetOrderDetail FROM
'SELECT orderID, Quantity, totalCost
 FROM Orders
 WHERE customerID = ?';

SET @CustomerID  = 2;
EXECUTE GetOrderDetail USING @CustomerID ;
```

```
orderID  Quantity  totalCost
2        2         250.00
```

```
#    Time      Action                                                                                      Message                                 Duration / Fetch
102  14:04:11  SELECT c.customerID,c.fullName,o.orderID,o.totalCost AS Cost,m.menuName,m_i.courseName, m_i.starterNa...  2 row(s) returned    0.000 sec / 0.000 sec
103  14:12:17  select * from orders LIMIT 0, 1000                                                          4 row(s) returned                       0.000 sec / 0.000 sec
104  14:13:07  insert into orders values(5,5,5,2,250.00)                                                   1 row(s) affected                       0.016 sec
105  14:14:08  PREPARE GetOrderDetail FROM 'SELECT orderID, Quantity, totalCost  FROM Orders  WHERE customerID ...  0 row(s) affected Statement prepared   0.000 sec
106  14:14:11  SET @CustomerID  = 2                                                                        0 row(s) affected                       0.000 sec
107  14:14:13  EXECUTE GetOrderDetail USING @CustomerID                                                    1 row(s) returned                       0.000 sec / 0.000 sec
```

Create a stored procedure called CancelOrder to delete an order record based on the user input of the order id.



```sql
use littlelemondb;
CREATE TABLE OrderCancellations (
    id INT AUTO_INCREMENT PRIMARY KEY,
    orderID INT,
    confirmation VARCHAR(255)
);
DELIMITER //
CREATE PROCEDURE CancelOrder(IN id INT)
BEGIN
    DELETE FROM Orders WHERE orderID = id;
    INSERT INTO OrderCancellations (orderID, confirmation) VALUES (id, CONCAT('Order ', id, ' is cancelled'));
    SELECT * FROM OrderCancellations;
END //
DELIMITER ;
Call CancelOrder(5);
```

```
#   Time      Action                                                                   Message                                                                                     Duration / Fetch
92  13:39:53  CREATE PROCEDURE CancelOrder(IN id INT) BEGIN   DELETE FROM Orders WHERE orderID = id;   IN...  0 row(s) affected                                                        0.000 sec
93  13:39:57  Call CancelOrder(5)                                                      2 row(s) returned                                                                           0.016 sec / 0.000 sec
94  13:42:27  delete from OrderCancellations;                                          Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses...  0.000 sec
95  13:43:01  select * from OrderCancellations LIMIT 0, 1000 ;                         2 row(s) returned                                                                           0.000 sec / 0.000 sec
96  13:43:15  delete from OrderCancellations where id = 1;                             1 row(s) affected                                                                           0.000 sec
97  13:43:18  delete from OrderCancellations where id = 2;                             1 row(s) affected                                                                           0.000 sec
```

Query Completed