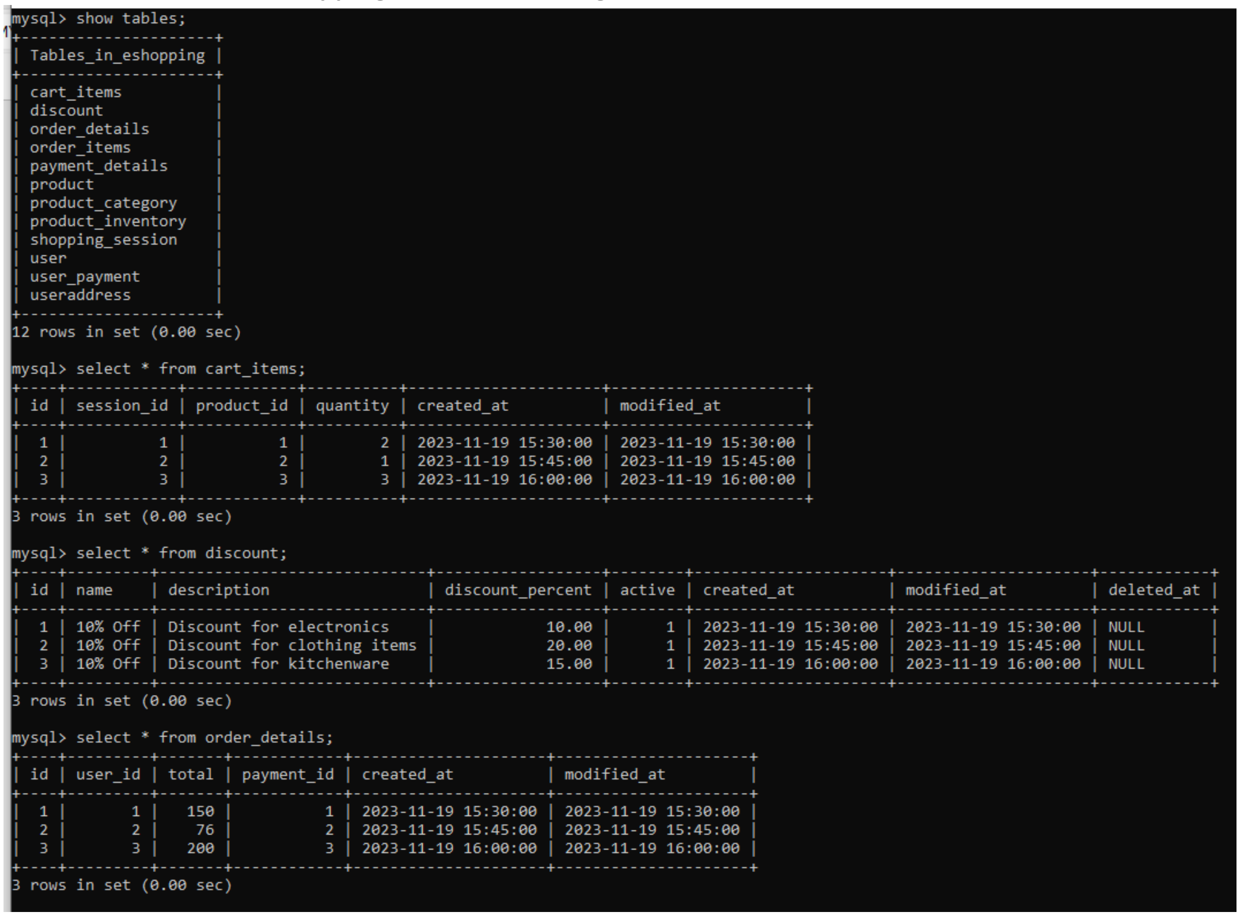
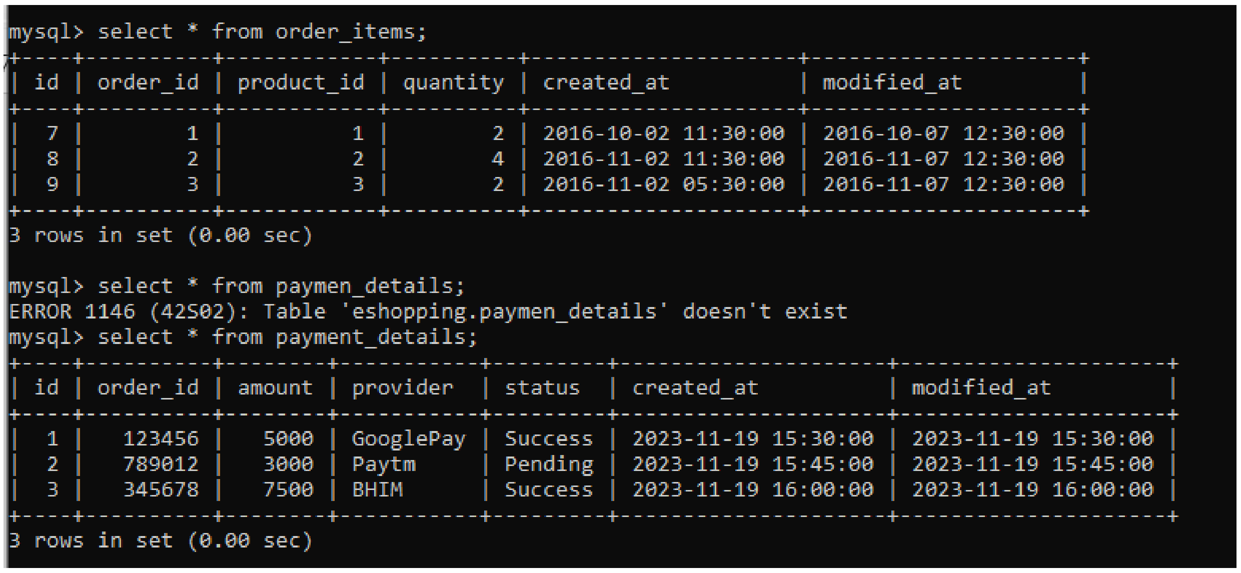
Lab Assignment 6 [DBMS PRACTICE QUESTION]

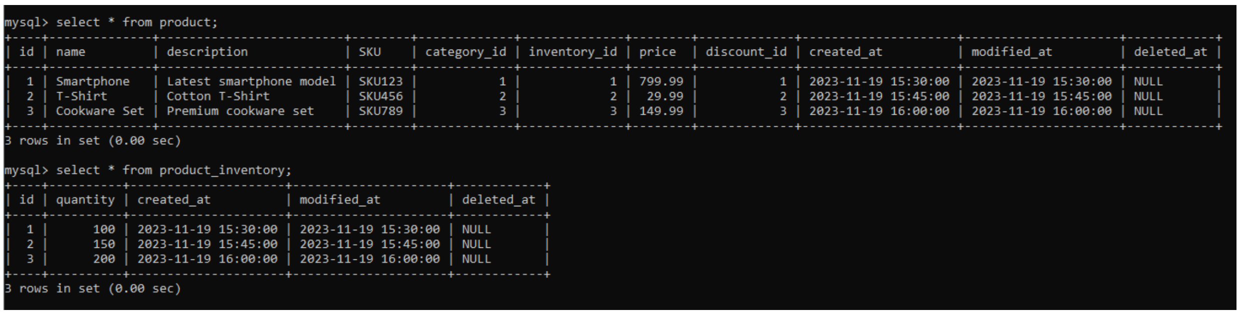
1. Use the database "eShopping" do the following



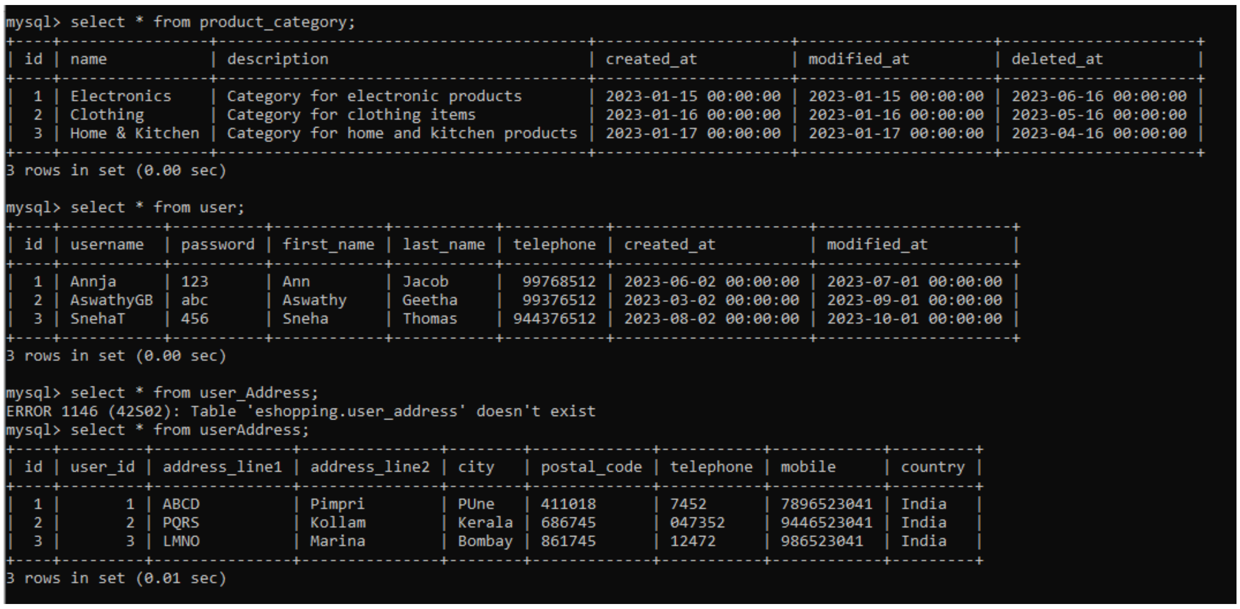
```
mysql> show tables;
+---------------------+
| Tables_in_eshopping |
+---------------------+
| cart_items          |
| discount            |
| order_details       |
| order_items         |
| payment_details     |
| product             |
| product_category    |
| product_inventory   |
| shopping_session    |
| user                |
| user_payment        |
| useraddress         |
+---------------------+
12 rows in set (0.00 sec)

mysql> select * from cart_items;
+----+------------+------------+----------+---------------------+---------------------+
| id | session_id | product_id | quantity | created_at          | modified_at         |
+----+------------+------------+----------+---------------------+---------------------+
|  1 |          1 |          1 |        2 | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 |
|  2 |          2 |          2 |        1 | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 |
|  3 |          3 |          3 |        3 | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 |
+----+------------+------------+----------+---------------------+---------------------+
3 rows in set (0.00 sec)

mysql> select * from discount;
+----+---------+-----------------------------+------------------+--------+---------------------+---------------------+------------+
| id | name    | description                 | discount_percent | active | created_at          | modified_at         | deleted_at |
+----+---------+-----------------------------+------------------+--------+---------------------+---------------------+------------+
|  1 | 10% Off | Discount for electronics    |            10.00 |      1 | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 | NULL       |
|  2 | 10% Off | Discount for clothing items |            20.00 |      1 | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 | NULL       |
|  3 | 10% Off | Discount for kitchenware    |            15.00 |      1 | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 | NULL       |
+----+---------+-----------------------------+------------------+--------+---------------------+---------------------+------------+
3 rows in set (0.00 sec)

mysql> select * from order_details;
+----+---------+-------+------------+---------------------+---------------------+
| id | user_id | total | payment_id | created_at          | modified_at         |
+----+---------+-------+------------+---------------------+---------------------+
|  1 |       1 |   150 |          1 | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 |
|  2 |       2 |    76 |          2 | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 |
|  3 |       3 |   200 |          3 | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 |
+----+---------+-------+------------+---------------------+---------------------+
3 rows in set (0.00 sec)
```



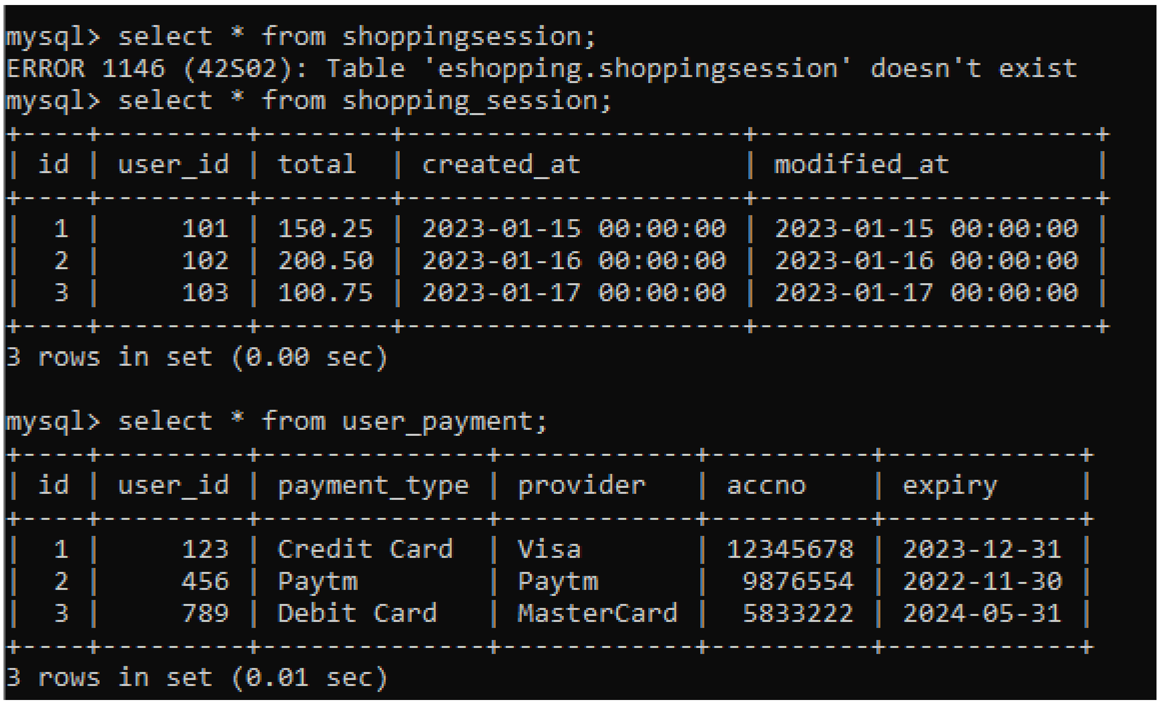
```
mysql> select * from order_items;
+----+----------+------------+----------+---------------------+---------------------+
| id | order_id | product_id | quantity | created_at          | modified_at         |
+----+----------+------------+----------+---------------------+---------------------+
|  7 |        1 |          1 |        2 | 2016-10-02 11:30:00 | 2016-10-07 12:30:00 |
|  8 |        2 |          2 |        4 | 2016-11-02 11:30:00 | 2016-11-07 12:30:00 |
|  9 |        3 |          3 |        2 | 2016-11-02 05:30:00 | 2016-11-07 12:30:00 |
+----+----------+------------+----------+---------------------+---------------------+
3 rows in set (0.00 sec)

mysql> select * from paymen_details;
ERROR 1146 (42S02): Table 'eshopping.paymen_details' doesn't exist
mysql> select * from payment_details;
+----+----------+--------+-----------+---------+---------------------+---------------------+
| id | order_id | amount | provider  | status  | created_at          | modified_at         |
+----+----------+--------+-----------+---------+---------------------+---------------------+
|  1 |   123456 |   5000 | GooglePay | Success | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 |
|  2 |   789012 |   3000 | Paytm     | Pending | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 |
|  3 |   345678 |   7500 | BHIM      | Success | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 |
+----+----------+--------+-----------+---------+---------------------+---------------------+
3 rows in set (0.00 sec)
```



```
mysql> select * from product;
+----+--------------+-------------------------+--------+-------------+--------------+--------+-------------+---------------------+---------------------+------------+
| id | name         | description             | SKU    | category_id | inventory_id | price  | discount_id | created_at          | modified_at         | deleted_at |
+----+--------------+-------------------------+--------+-------------+--------------+--------+-------------+---------------------+---------------------+------------+
|  1 | Smartphone   | Latest smartphone model | SKU123 |           1 |            1 | 799.99 |           1 | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 | NULL       |
|  2 | T-Shirt      | Cotton T-Shirt          | SKU456 |           2 |            2 |  29.99 |           2 | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 | NULL       |
|  3 | Cookware Set | Premium cookware set    | SKU789 |           3 |            3 | 149.99 |           3 | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 | NULL       |
+----+--------------+-------------------------+--------+-------------+--------------+--------+-------------+---------------------+---------------------+------------+
3 rows in set (0.00 sec)

mysql> select * from product_inventory;
+----+----------+---------------------+---------------------+------------+
| id | quantity | created_at          | modified_at         | deleted_at |
+----+----------+---------------------+---------------------+------------+
|  1 |      100 | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 | NULL       |
|  2 |      150 | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 | NULL       |
|  3 |      200 | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 | NULL       |
+----+----------+---------------------+---------------------+------------+
3 rows in set (0.00 sec)
```

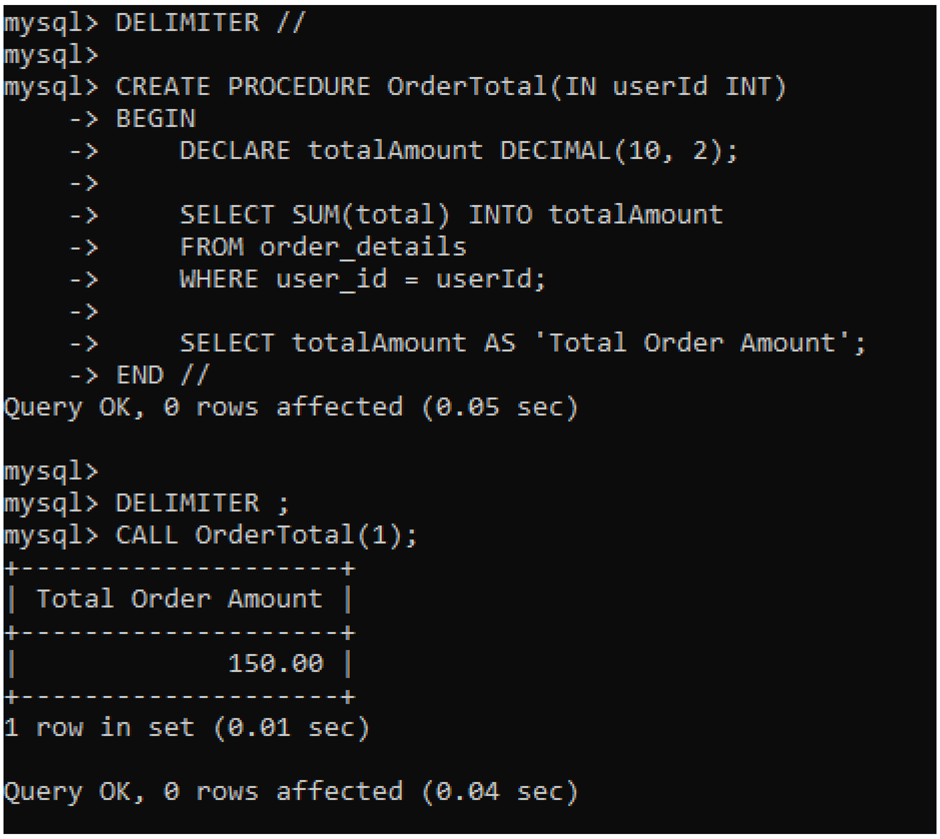


```
mysql> select * from product_category;
+----+----------------+----------------------------------------+---------------------+---------------------+---------------------+
| id | name           | description                            | created_at          | modified_at         | deleted_at          |
+----+----------------+----------------------------------------+---------------------+---------------------+---------------------+
|  1 | Electronics    | Category for electronic products       | 2023-01-15 00:00:00 | 2023-01-15 00:00:00 | 2023-06-16 00:00:00 |
|  2 | Clothing       | Category for clothing items            | 2023-01-16 00:00:00 | 2023-01-16 00:00:00 | 2023-05-16 00:00:00 |
|  3 | Home & Kitchen | Category for home and kitchen products | 2023-01-17 00:00:00 | 2023-01-17 00:00:00 | 2023-04-16 00:00:00 |
+----+----------------+----------------------------------------+---------------------+---------------------+---------------------+
3 rows in set (0.00 sec)

mysql> select * from user;
+----+-----------+----------+------------+-----------+-----------+---------------------+---------------------+
| id | username  | password | first_name | last_name | telephone | created_at          | modified_at         |
+----+-----------+----------+------------+-----------+-----------+---------------------+---------------------+
|  1 | Annja     | 123      | Ann        | Jacob     |  99768512 | 2023-06-02 00:00:00 | 2023-07-01 00:00:00 |
|  2 | AswathyGB | abc      | Aswathy    | Geetha    |  99376512 | 2023-03-02 00:00:00 | 2023-09-01 00:00:00 |
|  3 | SnehaT    | 456      | Sneha      | Thomas    | 944376512 | 2023-08-02 00:00:00 | 2023-10-01 00:00:00 |
+----+-----------+----------+------------+-----------+-----------+---------------------+---------------------+
3 rows in set (0.00 sec)

mysql> select * from user_Address;
ERROR 1146 (42S02): Table 'eshopping.user_address' doesn't exist
mysql> select * from userAddress;
+----+---------+---------------+---------------+--------+-------------+-----------+------------+---------+
| id | user_id | address_line1 | address_line2 | city   | postal_code | telephone | mobile     | country |
+----+---------+---------------+---------------+--------+-------------+-----------+------------+---------+
|  1 |       1 | ABCD          | Pimpri        | PUne   | 411018      | 7452      | 7896523041 | India   |
|  2 |       2 | PQRS          | Kollam        | Kerala | 686745      | 047352    | 9446523041 | India   |
|  3 |       3 | LMNO          | Marina        | Bombay | 861745      | 12472     | 986523041  | India   |
+----+---------+---------------+---------------+--------+-------------+-----------+------------+---------+
3 rows in set (0.01 sec)
```



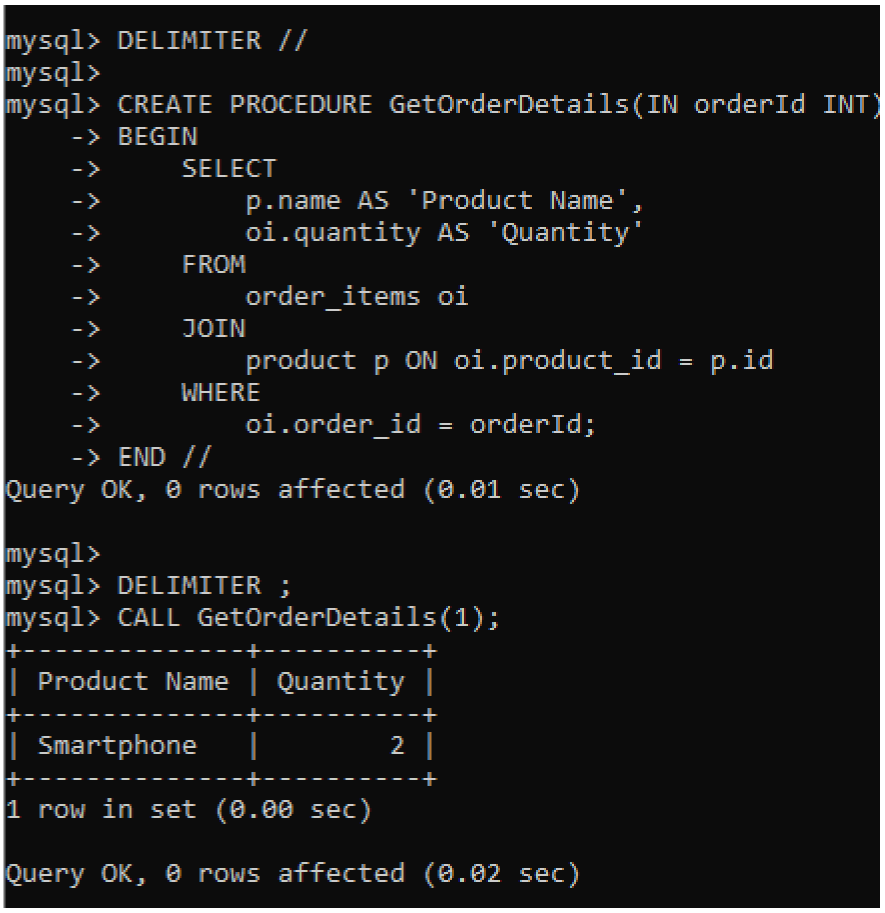
```
mysql> select * from shoppingsession;
ERROR 1146 (42S02): Table 'eshopping.shoppingsession' doesn't exist
mysql> select * from shopping_session;
+----+---------+--------+---------------------+---------------------+
| id | user_id | total  | created_at          | modified_at         |
+----+---------+--------+---------------------+---------------------+
|  1 |     101 | 150.25 | 2023-01-15 00:00:00 | 2023-01-15 00:00:00 |
|  2 |     102 | 200.50 | 2023-01-16 00:00:00 | 2023-01-16 00:00:00 |
|  3 |     103 | 100.75 | 2023-01-17 00:00:00 | 2023-01-17 00:00:00 |
+----+---------+--------+---------------------+---------------------+
3 rows in set (0.00 sec)

mysql> select * from user_payment;
+----+---------+--------------+------------+----------+------------+
| id | user_id | payment_type | provider   | accno    | expiry     |
+----+---------+--------------+------------+----------+------------+
|  1 |     123 | Credit Card  | Visa       | 12345678 | 2023-12-31 |
|  2 |     456 | Paytm        | Paytm      |  9876554 | 2022-11-30 |
|  3 |     789 | Debit Card   | MasterCard |  5833222 | 2024-05-31 |
+----+---------+--------------+------------+----------+------------+
3 rows in set (0.01 sec)
```

a. Write a stored procedure, named OrderTotal() to return the sum of all order total amount for a user. Pass user id as the input to the procedure. (use order_details table for this)



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE OrderTotal(IN userId INT)
    -> BEGIN
    ->     DECLARE totalAmount DECIMAL(10, 2);
    ->
    ->     SELECT SUM(total) INTO totalAmount
    ->     FROM order_details
    ->     WHERE user_id = userId;
    ->
    ->     SELECT totalAmount AS 'Total Order Amount';
    -> END //
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL OrderTotal(1);
+--------------------+
| Total Order Amount |
+--------------------+
|             150.00 |
+--------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.04 sec)
```

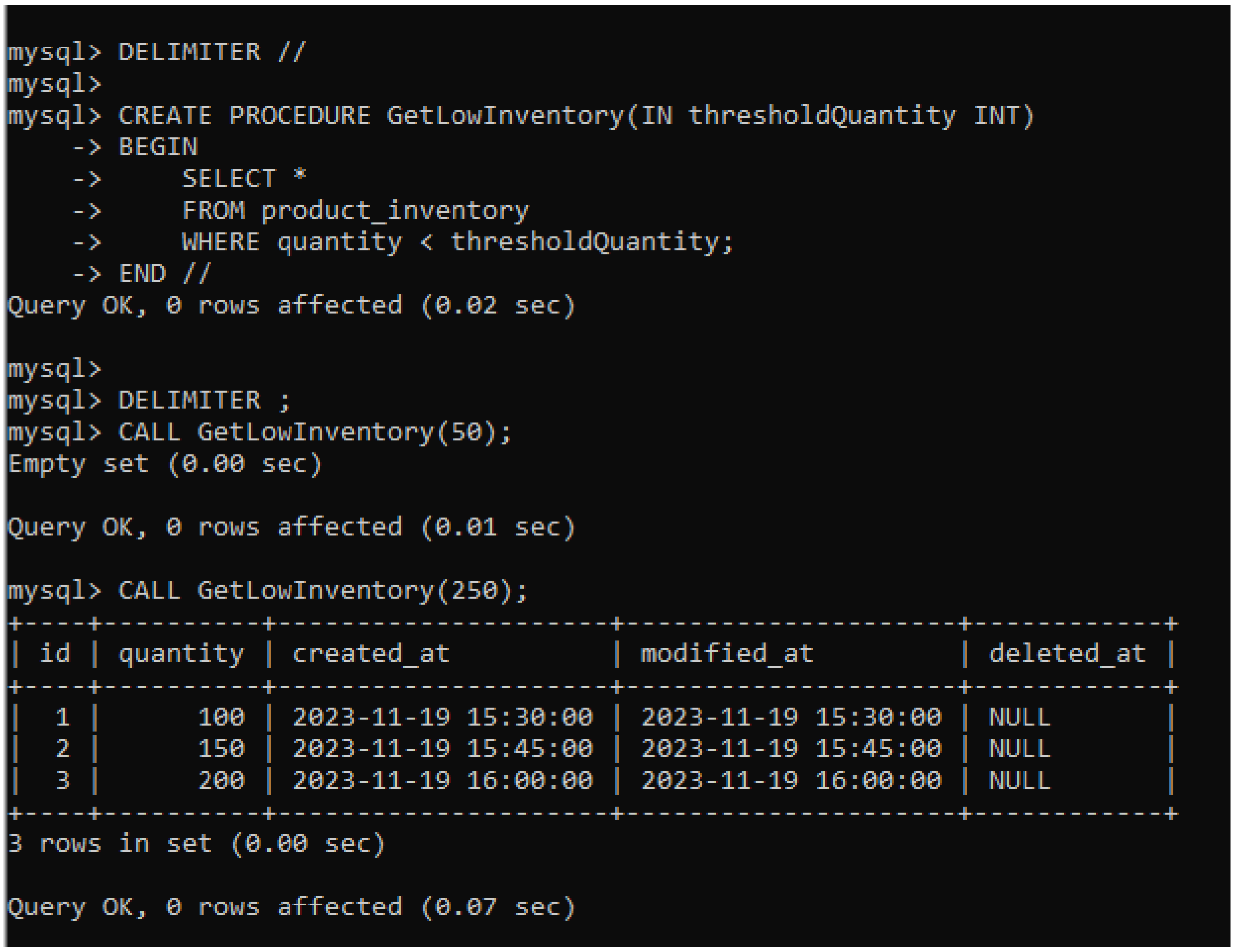
b. Create a stored procedure that takes orderid as the input and display all the products, quantity of each product in it.



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE GetOrderDetails(IN orderId INT)
    -> BEGIN
    ->     SELECT
    ->         p.name AS 'Product Name',
    ->         oi.quantity AS 'Quantity'
    ->     FROM
    ->         order_items oi
    ->     JOIN
    ->         product p ON oi.product_id = p.id
    ->     WHERE
    ->         oi.order_id = orderId;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetOrderDetails(1);
+--------------+----------+
| Product Name | Quantity |
+--------------+----------+
| Smartphone   |        2 |
+--------------+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

c. Develop a stored procedure that retrieves records from product_inventory where

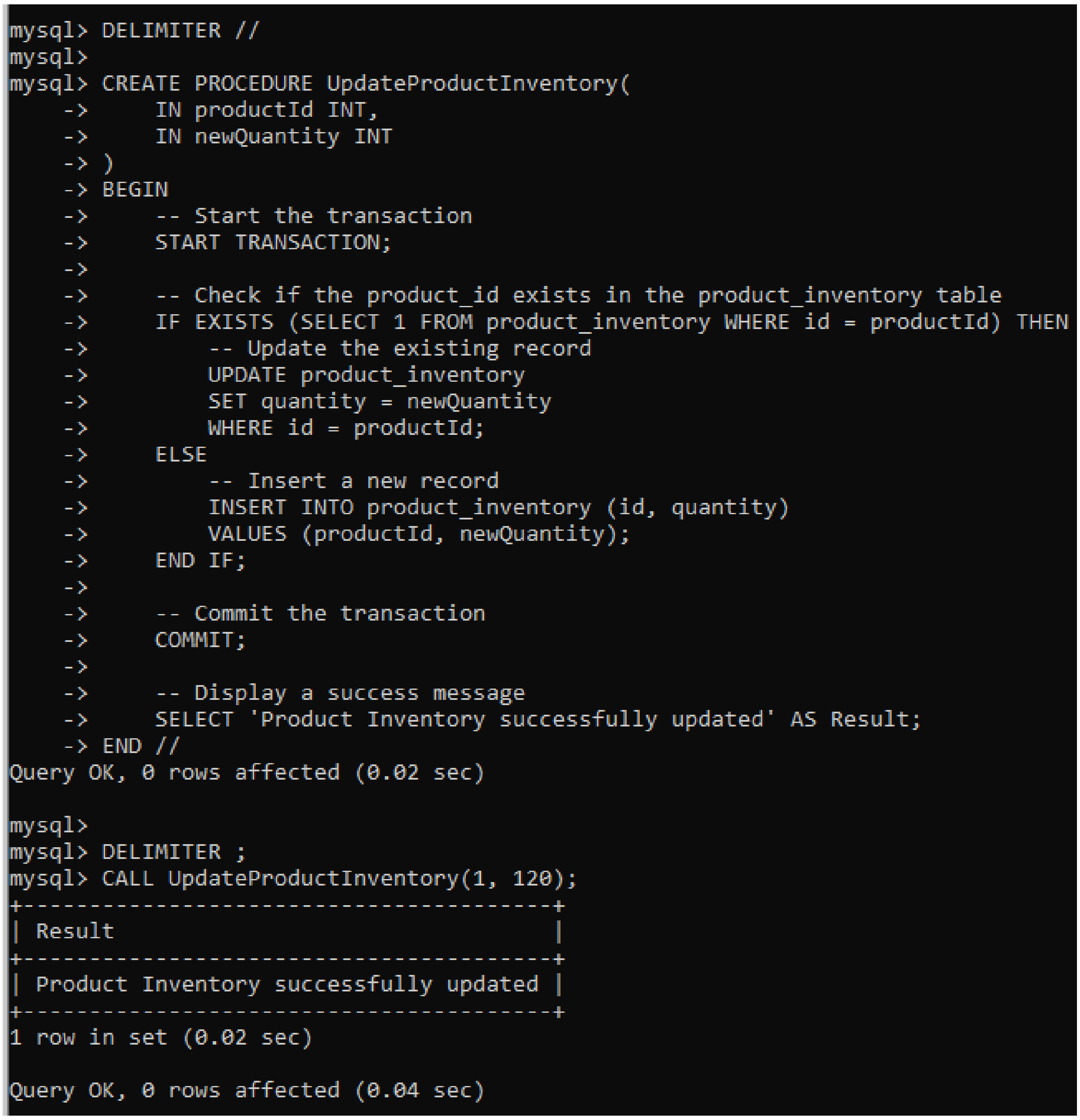
the quantity is less than a provided value(input to the procedure)



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE GetLowInventory(IN thresholdQuantity INT)
    -> BEGIN
    ->     SELECT *
    ->     FROM product_inventory
    ->     WHERE quantity < thresholdQuantity;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetLowInventory(50);
Empty set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL GetLowInventory(250);
+----+----------+---------------------+---------------------+------------+
| id | quantity | created_at          | modified_at         | deleted_at |
+----+----------+---------------------+---------------------+------------+
|  1 |      100 | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 | NULL       |
|  2 |      150 | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 | NULL       |
|  3 |      200 | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 | NULL       |
+----+----------+---------------------+---------------------+------------+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.07 sec)
```

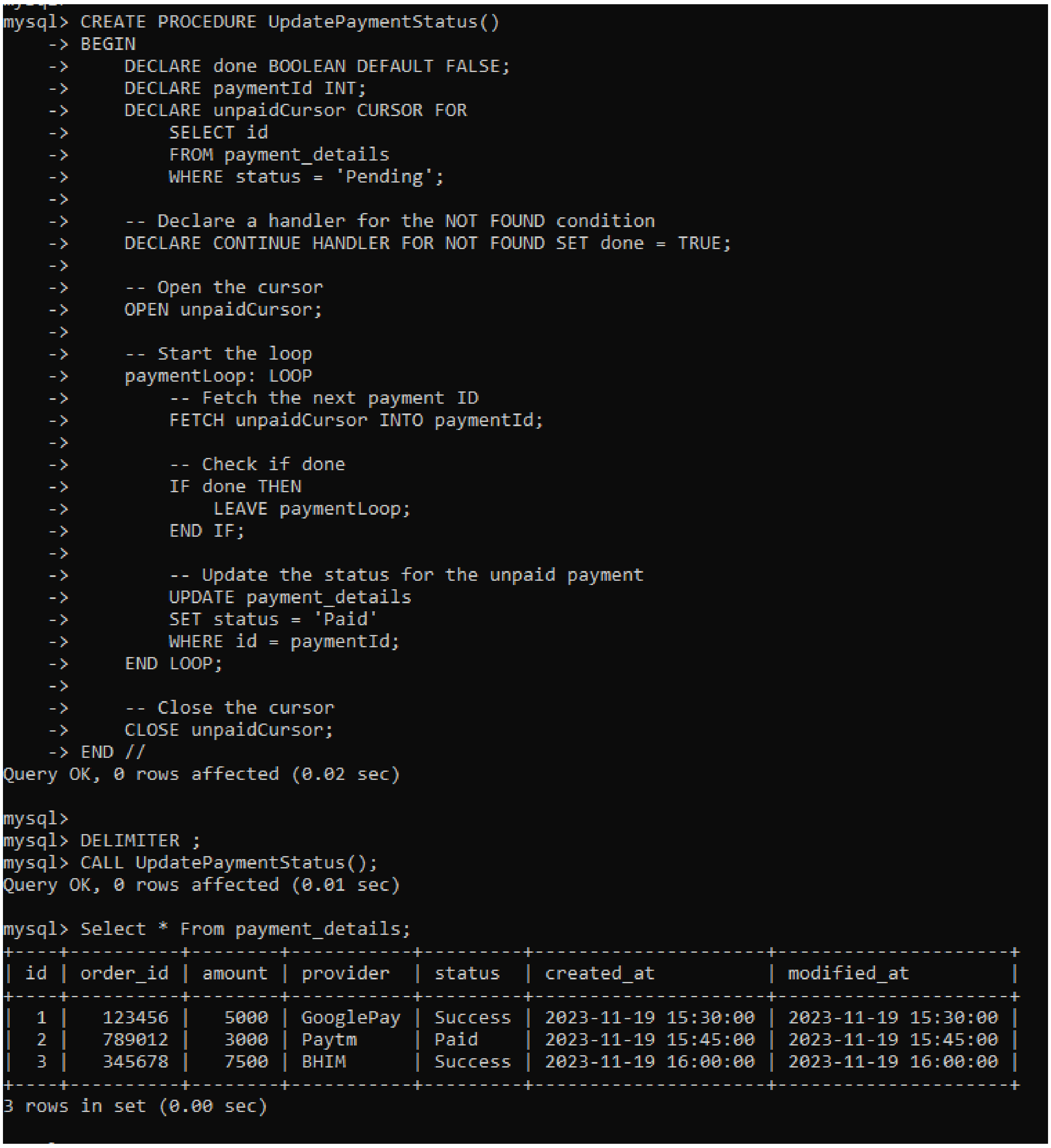
d. Create a stored procedure that updates a record in the product_inventory table, if

it exists; otherwise, inserts a new record for a particular productid.



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE UpdateProductInventory(
    ->     IN productId INT,
    ->     IN newQuantity INT
    -> )
    -> BEGIN
    ->     -- Start the transaction
    ->     START TRANSACTION;
    ->
    ->     -- Check if the product_id exists in the product_inventory table
    ->     IF EXISTS (SELECT 1 FROM product_inventory WHERE id = productId) THEN
    ->         -- Update the existing record
    ->         UPDATE product_inventory
    ->         SET quantity = newQuantity
    ->         WHERE id = productId;
    ->     ELSE
    ->         -- Insert a new record
    ->         INSERT INTO product_inventory (id, quantity)
    ->         VALUES (productId, newQuantity);
    ->     END IF;
    ->
    ->     -- Commit the transaction
    ->     COMMIT;
    ->
    ->     -- Display a success message
    ->     SELECT 'Product Inventory successfully updated' AS Result;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL UpdateProductInventory(1, 120);
+----------------------------------------+
| Result                                 |
+----------------------------------------+
| Product Inventory successfully updated |
+----------------------------------------+
1 row in set (0.02 sec)

Query OK, 0 rows affected (0.04 sec)
```

e. Develop a stored procedure that uses a cursor to loop to iterate through a set of records of payment_details table and performs status updation for every unpaid payments.

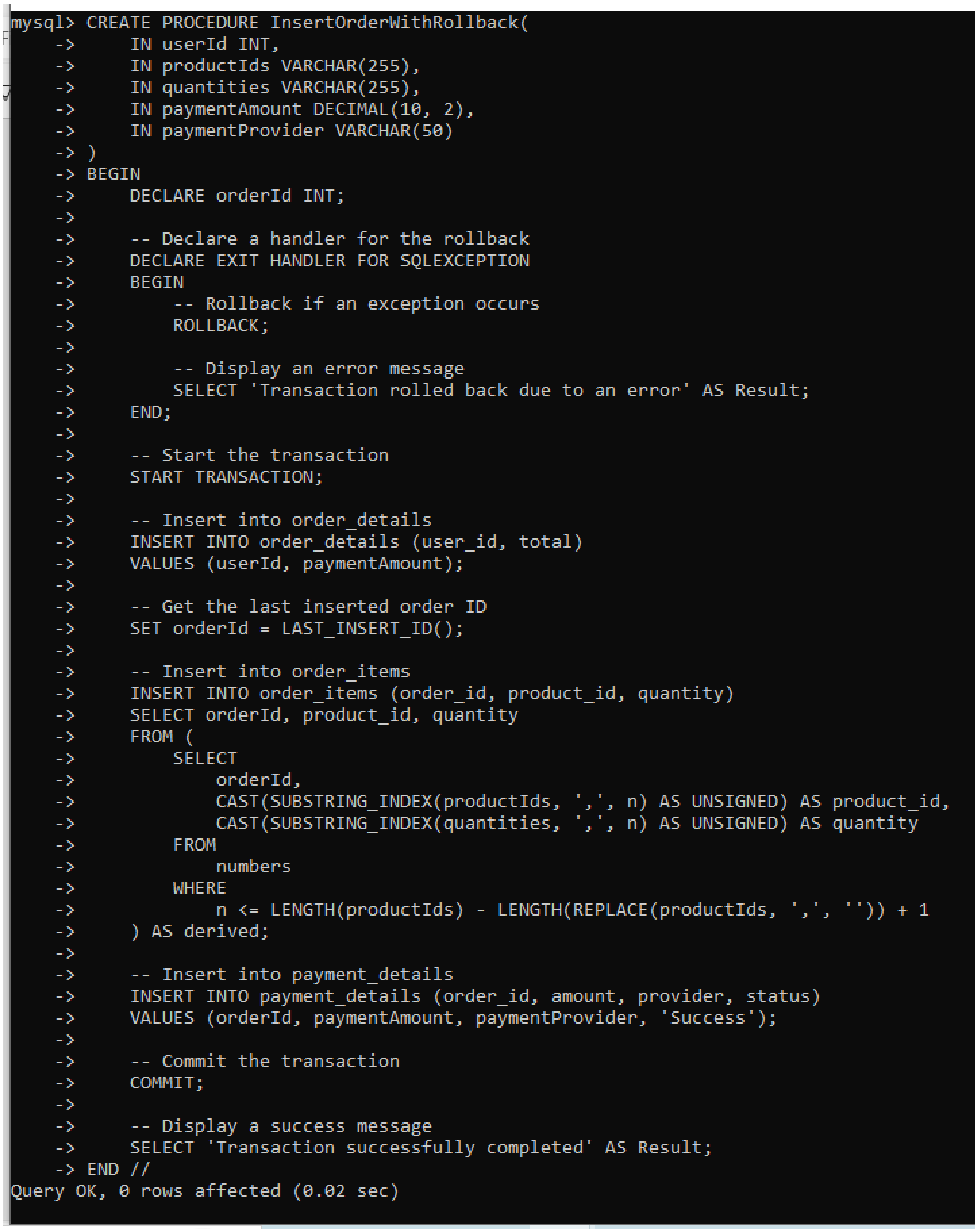


```
mysql> CREATE PROCEDURE UpdatePaymentStatus()
    -> BEGIN
    ->     DECLARE done BOOLEAN DEFAULT FALSE;
    ->     DECLARE paymentId INT;
    ->     DECLARE unpaidCursor CURSOR FOR
    ->         SELECT id
    ->         FROM payment_details
    ->         WHERE status = 'Pending';
    ->
    ->     -- Declare a handler for the NOT FOUND condition
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     -- Open the cursor
    ->     OPEN unpaidCursor;
    ->
    ->     -- Start the loop
    ->     paymentLoop: LOOP
    ->         -- Fetch the next payment ID
    ->         FETCH unpaidCursor INTO paymentId;
    ->
    ->         -- Check if done
    ->         IF done THEN
    ->             LEAVE paymentLoop;
    ->         END IF;
    ->
    ->         -- Update the status for the unpaid payment
    ->         UPDATE payment_details
    ->         SET status = 'Paid'
    ->         WHERE id = paymentId;
    ->     END LOOP;
    ->
    ->     -- Close the cursor
    ->     CLOSE unpaidCursor;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL UpdatePaymentStatus();
Query OK, 0 rows affected (0.01 sec)

mysql> Select * From payment_details;
+----+----------+--------+-----------+---------+---------------------+---------------------+
| id | order_id | amount | provider  | status  | created_at          | modified_at         |
+----+----------+--------+-----------+---------+---------------------+---------------------+
|  1 |   123456 |   5000 | GooglePay | Success | 2023-11-19 15:30:00 | 2023-11-19 15:30:00 |
|  2 |   789012 |   3000 | Paytm     | Paid    | 2023-11-19 15:45:00 | 2023-11-19 15:45:00 |
|  3 |   345678 |   7500 | BHIM      | Success | 2023-11-19 16:00:00 | 2023-11-19 16:00:00 |
+----+----------+--------+-----------+---------+---------------------+---------------------+
3 rows in set (0.00 sec)
```

f. Implement a stored procedure that rolls back all transactions if the following

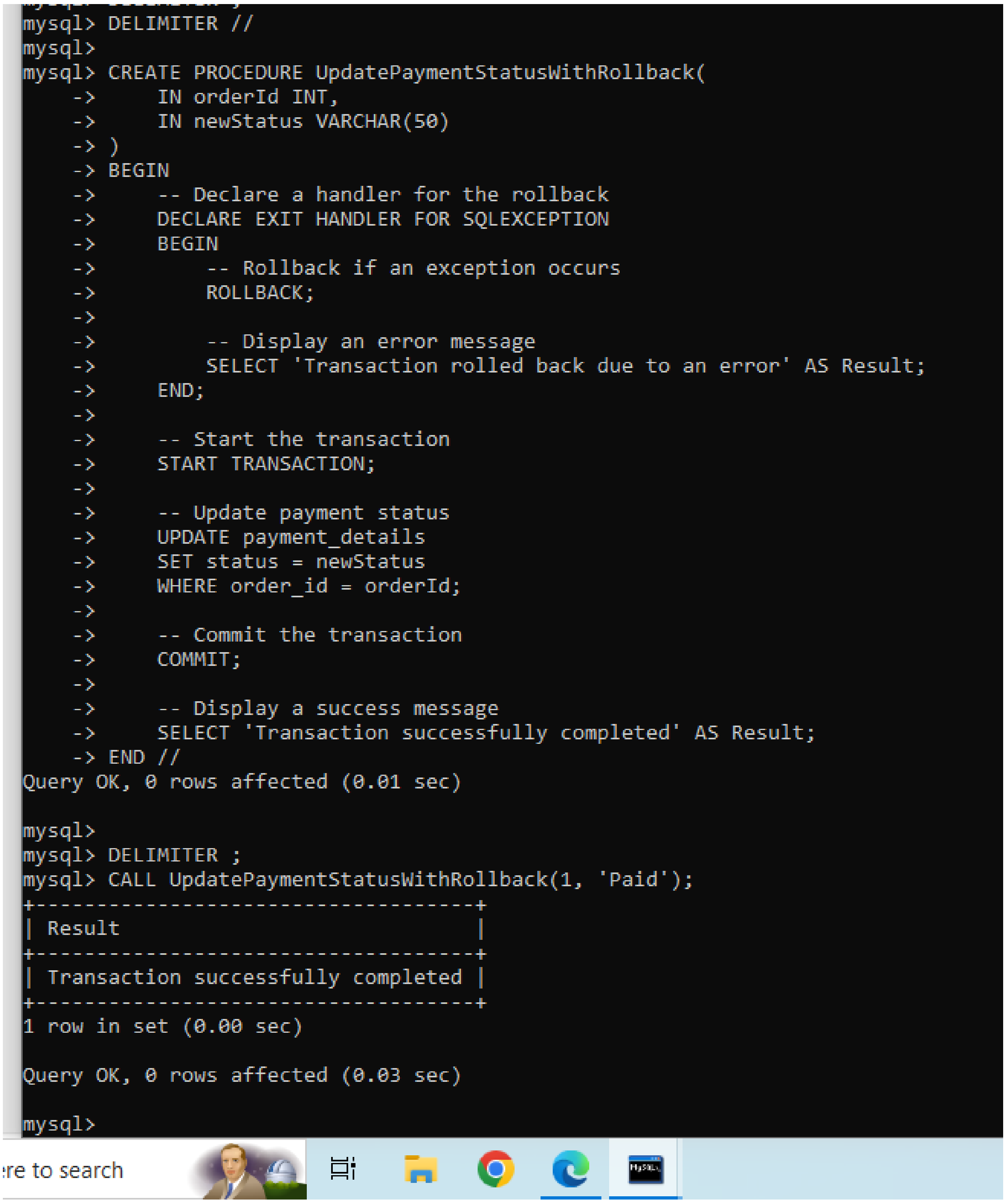
conditions are not met.

i. Insert data into order_details, order-items, payment_details tables.



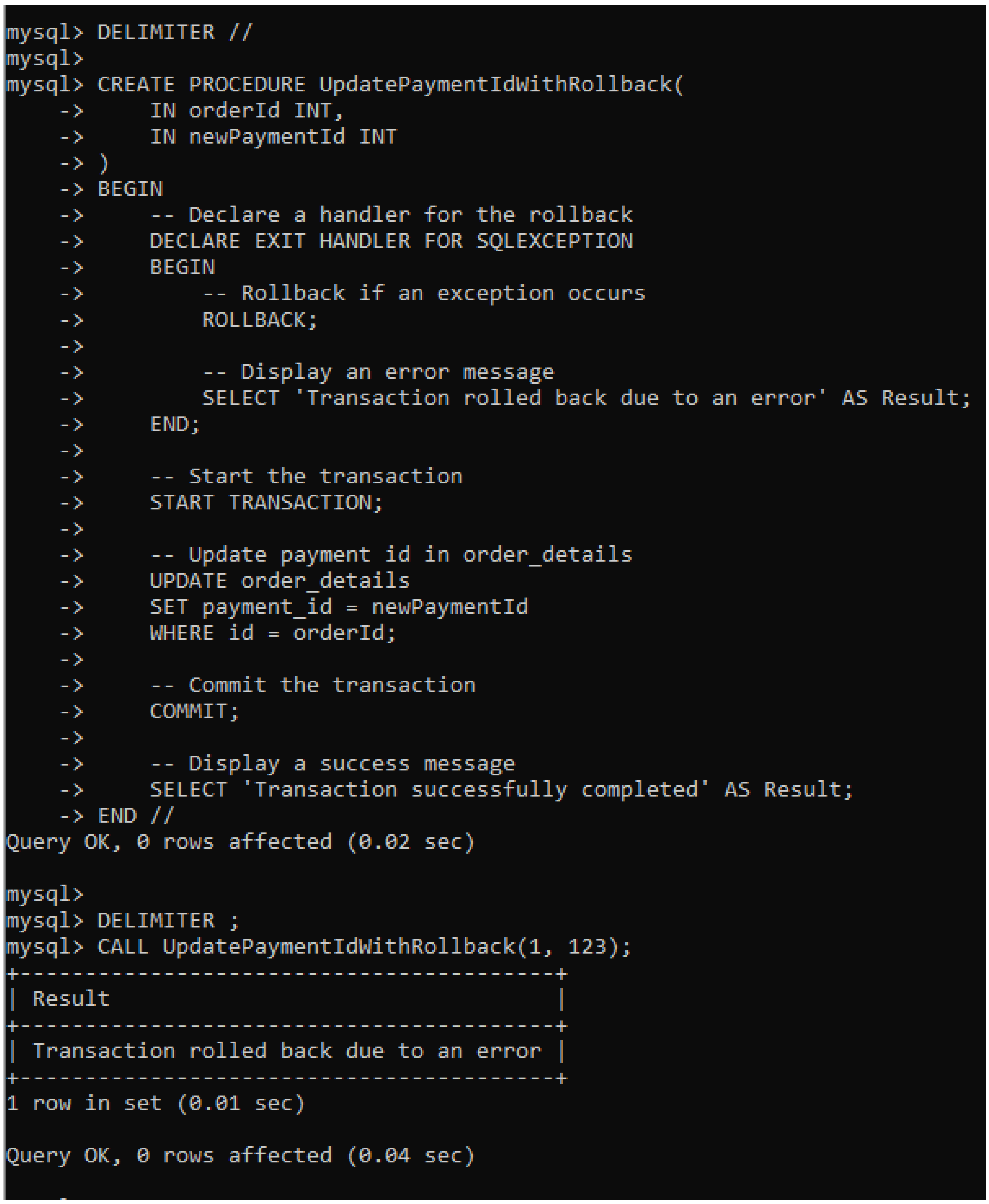
```
mysql> CREATE PROCEDURE InsertOrderWithRollback(
    ->     IN userId INT,
    ->     IN productIds VARCHAR(255),
    ->     IN quantities VARCHAR(255),
    ->     IN paymentAmount DECIMAL(10, 2),
    ->     IN paymentProvider VARCHAR(50)
    -> )
    -> BEGIN
    ->     DECLARE orderId INT;
    ->
    ->     -- Declare a handler for the rollback
    ->     DECLARE EXIT HANDLER FOR SQLEXCEPTION
    ->     BEGIN
    ->         -- Rollback if an exception occurs
    ->         ROLLBACK;
    ->
    ->         -- Display an error message
    ->         SELECT 'Transaction rolled back due to an error' AS Result;
    ->     END;
    ->
    ->     -- Start the transaction
    ->     START TRANSACTION;
    ->
    ->     -- Insert into order_details
    ->     INSERT INTO order_details (user_id, total)
    ->     VALUES (userId, paymentAmount);
    ->
    ->     -- Get the last inserted order ID
    ->     SET orderId = LAST_INSERT_ID();
    ->
    ->     -- Insert into order_items
    ->     INSERT INTO order_items (order_id, product_id, quantity)
    ->     SELECT orderId, product_id, quantity
    ->     FROM (
    ->         SELECT
    ->             orderId,
    ->             CAST(SUBSTRING_INDEX(productIds, ',', n) AS UNSIGNED) AS product_id,
    ->             CAST(SUBSTRING_INDEX(quantities, ',', n) AS UNSIGNED) AS quantity
    ->         FROM
    ->             numbers
    ->         WHERE
    ->             n <= LENGTH(productIds) - LENGTH(REPLACE(productIds, ',', '')) + 1
    ->     ) AS derived;
    ->
    ->     -- Insert into payment_details
    ->     INSERT INTO payment_details (order_id, amount, provider, status)
    ->     VALUES (orderId, paymentAmount, paymentProvider, 'Success');
    ->
    ->     -- Commit the transaction
    ->     COMMIT;
    ->
    ->     -- Display a success message
    ->     SELECT 'Transaction successfully completed' AS Result;
    -> END //
Query OK, 0 rows affected (0.02 sec)
```

ii. Update patment_details table to set status as paid.



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE UpdatePaymentStatusWithRollback(
    ->     IN orderId INT,
    ->     IN newStatus VARCHAR(50)
    -> )
    -> BEGIN
    ->     -- Declare a handler for the rollback
    ->     DECLARE EXIT HANDLER FOR SQLEXCEPTION
    ->     BEGIN
    ->         -- Rollback if an exception occurs
    ->         ROLLBACK;
    ->
    ->         -- Display an error message
    ->         SELECT 'Transaction rolled back due to an error' AS Result;
    ->     END;
    ->
    ->     -- Start the transaction
    ->     START TRANSACTION;
    ->
    ->     -- Update payment status
    ->     UPDATE payment_details
    ->     SET status = newStatus
    ->     WHERE order_id = orderId;
    ->
    ->     -- Commit the transaction
    ->     COMMIT;
    ->
    ->     -- Display a success message
    ->     SELECT 'Transaction successfully completed' AS Result;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL UpdatePaymentStatusWithRollback(1, 'Paid');
+------------------------------------+
| Result                             |
+------------------------------------+
| Transaction successfully completed |
+------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.03 sec)

mysql>
```

iii. Update payment id in the order_details table.



```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE UpdatePaymentIdWithRollback(
    ->     IN orderId INT,
    ->     IN newPaymentId INT
    -> )
    -> BEGIN
    ->     -- Declare a handler for the rollback
    ->     DECLARE EXIT HANDLER FOR SQLEXCEPTION
    ->     BEGIN
    ->         -- Rollback if an exception occurs
    ->         ROLLBACK;
    ->
    ->         -- Display an error message
    ->         SELECT 'Transaction rolled back due to an error' AS Result;
    ->     END;
    ->
    ->     -- Start the transaction
    ->     START TRANSACTION;
    ->
    ->     -- Update payment id in order_details
    ->     UPDATE order_details
    ->     SET payment_id = newPaymentId
    ->     WHERE id = orderId;
    ->
    ->     -- Commit the transaction
    ->     COMMIT;
    ->
    ->     -- Display a success message
    ->     SELECT 'Transaction successfully completed' AS Result;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL UpdatePaymentIdWithRollback(1, 123);
+-----------------------------------------+
| Result                                  |
+-----------------------------------------+
| Transaction rolled back due to an error |
+-----------------------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.04 sec)
```