Assignment No. 6

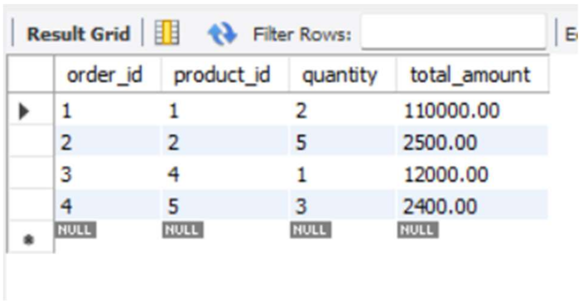
-Yash Yadav(107)

Question 1: Update the total_amount in the orders table using the corresponding product price from the products table.

Code:-

```
SET SQL_SAFE_UPDATES = 0;

UPDATE orders o

JOIN products p ON o.product_id = p.product_id

SET o.total_amount = o.quantity * p.price;

SET SQL_SAFE_UPDATES = 1;

SELECT * FROM orders;
```

Output:-



order_id	product_id	quantity	total_amount
1	1	2	110000.00
2	2	5	2500.00
3	4	1	12000.00
4	5	3	2400.00
NULL	NULL	NULL	NULL

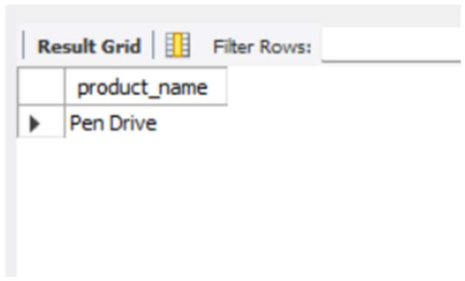
Question 2: Fetch all product names from the products table using a cursor and display them.

Code:-

```
DELIMITER //

CREATE PROCEDURE show_product_names()

BEGIN

   DECLARE done INT DEFAULT 0;
```

```
    DECLARE pname VARCHAR(100);

    DECLARE cur CURSOR FOR SELECT product_name FROM products;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    read_loop: LOOP

        FETCH cur INTO pname;

        IF done THEN

            LEAVE read_loop;

        END IF;

        SELECT pname AS product_name;

    END LOOP;

    CLOSE cur;

END //

DELIMITER ;

CALL show_product_names();
```

Output:-



```
product_name
Pen Drive
```

Question 3: Copy all data from the orders table into the order_audit table using a cursor.

Code:-

```
DROP PROCEDURE IF EXISTS copy_orders_to_audit;
```

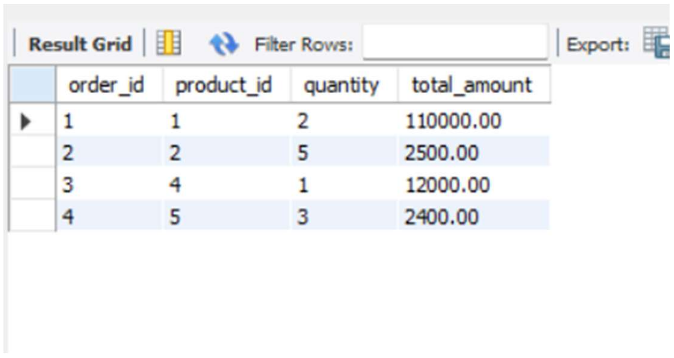
```
DELIMITER //

CREATE PROCEDURE copy_orders_to_audit()

BEGIN

    DECLARE done INT DEFAULT 0;

    DECLARE oid INT;

    DECLARE pid INT;

    DECLARE qty INT;

    DECLARE amt DECIMAL(10,2);

    DECLARE cur CURSOR FOR

        SELECT order_id, product_id, quantity, total_amount FROM orders;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    read_loop: LOOP

        FETCH cur INTO oid, pid, qty, amt;

        IF done THEN

            LEAVE read_loop;

        END IF;

        INSERT INTO order_audit (order_id, product_id, quantity, total_amount)

        VALUES (oid, pid, qty, amt);

    END LOOP;

    CLOSE cur;

END //
```

DELIMITER ;

CALL copy_orders_to_audit();

SELECT * FROM order_audit;

Output:-


Result Grid | Filter Rows: | Export:

order_id	product_id	quantity	total_amount
1	1	2	110000.00
2	2	5	2500.00
3	4	1	12000.00
4	5	3	2400.00

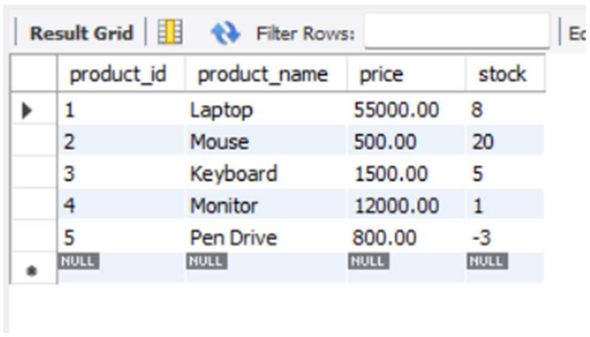
Question 4: Reduce the stock in the products table for each order processed based on the ordered quantity.

Code:-

```
DELIMITER //

CREATE PROCEDURE reduce_product_stock()

BEGIN

  DECLARE done INT DEFAULT 0;

  DECLARE oid INT;

  DECLARE pid INT;

  DECLARE qty INT;

  DECLARE cur CURSOR FOR

    SELECT order_id, product_id, quantity FROM orders;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cur;

  read_loop: LOOP

    FETCH cur INTO oid, pid, qty;
```

```
        IF done THEN
            LEAVE read_loop;
        END IF;
        UPDATE products
        SET stock = stock - qty
        WHERE product_id = pid;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
CALL reduce_product_stock();
SELECT * FROM products;
```

Output:-



product_id	product_name	price	stock
1	Laptop	55000.00	8
2	Mouse	500.00	20
3	Keyboard	1500.00	5
4	Monitor	12000.00	1
5	Pen Drive	800.00	-3
NULL	NULL	NULL	NULL

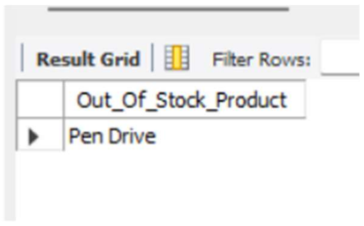
Question 5: Display all product names that are out of stock using a cursor.

Code:-

```
DELIMITER //
CREATE PROCEDURE show_out_of_stock()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pname VARCHAR(100);
    DECLARE cur CURSOR FOR
```

```
        SELECT product_name FROM products WHERE stock <= 0;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO pname;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT pname AS Out_Of_Stock_Product;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
CALL show_out_of_stock();
```

Output:-



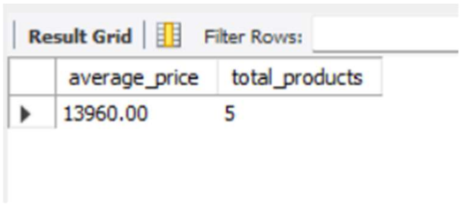
```
Out_Of_Stock_Product
Pen Drive
```

Question 6: Calculate and display the average price of all products using a cursor.

Code:-

```
DELIMITER //
CREATE PROCEDURE avg_product_price()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE v_price DECIMAL(10,2);
    DECLARE v_sum DECIMAL(16,2) DEFAULT 0;
    DECLARE v_count INT DEFAULT 0;
    DECLARE v_avg DECIMAL(16,2);
    DECLARE cur CURSOR FOR SELECT price FROM products;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO v_price;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SET v_sum = v_sum + v_price;
        SET v_count = v_count + 1;
    END LOOP;
    CLOSE cur;
    IF v_count = 0 THEN
        SELECT NULL AS average_price, 'no products' AS note;
    ELSE
        SET v_avg = v_sum / v_count;
        SELECT v_avg AS average_price, v_count AS total_products;
    END IF;
END //
DELIMITER ;
CALL avg_product_price();
```

Output:-


Result Grid | Filter Rows:

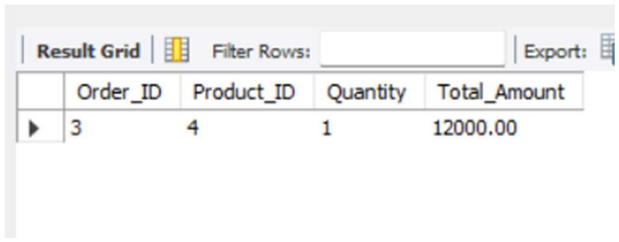
```
  average_price  total_products
▶ 13960.00       5
```

Question 7: Display all orders whose total amount is greater than ₹10,000 using a cursor.

Code:-

```
DELIMITER //

CREATE PROCEDURE show_high_value_orders()

BEGIN

  DECLARE done INT DEFAULT 0;

  DECLARE oid INT;

  DECLARE pid INT;

  DECLARE qty INT;

  DECLARE amt DECIMAL(10,2);

  DECLARE cur CURSOR FOR

    SELECT order_id, product_id, quantity, total_amount FROM orders;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN cur;

  read_loop: LOOP

    FETCH cur INTO oid, pid, qty, amt;

    IF done THEN

      LEAVE read_loop;

    END IF;

    IF amt > 10000 THEN

      SELECT oid AS Order_ID,

          pid AS Product_ID,
            qty AS Quantity,

            amt AS Total_Amount;

      END IF;

   END LOOP;

   CLOSE cur;

END //

DELIMITER ;

CALL show_high_value_orders();
```

Output:-



```
Result Grid | Filter Rows: | Export:
   Order_ID  Product_ID  Quantity  Total_Amount
▶  3         4           1         12000.00
```

Question 8: Create a summary table showing each product and its total quantity sold using a cursor.

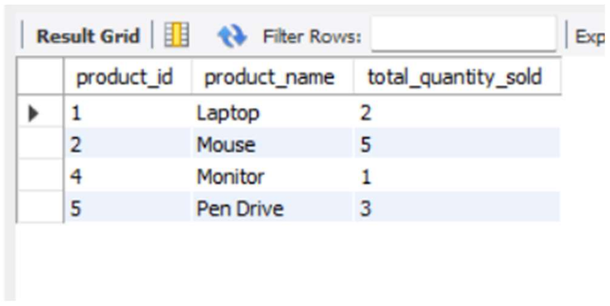
Code:-

```
CREATE TABLE IF NOT EXISTS product_summary (

   product_id INT PRIMARY KEY,

   total_quantity_sold INT

);


DELIMITER //

CREATE PROCEDURE create_product_summary()

BEGIN

   DECLARE done INT DEFAULT 0;

   DECLARE pid INT;

   DECLARE qty INT;
```

```
    DECLARE cur CURSOR FOR

        SELECT product_id, quantity FROM orders;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    TRUNCATE TABLE product_summary;

    OPEN cur;

    read_loop: LOOP

        FETCH cur INTO pid, qty;

        IF done THEN

            LEAVE read_loop;

        END IF;

        IF EXISTS (SELECT * FROM product_summary WHERE product_id = pid) THEN

            UPDATE product_summary

            SET total_quantity_sold = total_quantity_sold + qty

            WHERE product_id = pid;

        ELSE


            INSERT INTO product_summary (product_id, total_quantity_sold)

            VALUES (pid, qty);

        END IF;

    END LOOP;

    CLOSE cur;

END //

DELIMITER ;

CALL create_product_summary();


SELECT ps.product_id, p.product_name, ps.total_quantity_sold
```

```
FROM product_summary ps

JOIN products p ON ps.product_id = p.product_id;
```

Output:-



Result Grid | Filter Rows: | Exp

product_id	product_name	total_quantity_sold
1	Laptop	2
2	Mouse	5
4	Monitor	1
5	Pen Drive	3

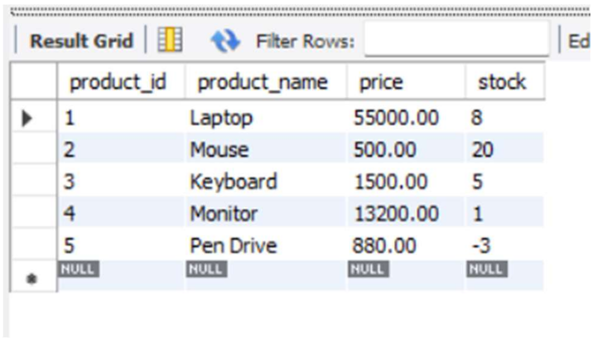
Question 9: Increase the price of all products with stock less than 5 by 10% using a cursor.

Code:-

```
DELIMITER //

CREATE PROCEDURE increase_price_low_stock()

BEGIN

   DECLARE done INT DEFAULT 0;

   DECLARE pid INT;

   DECLARE pstock INT;

   DECLARE pprice DECIMAL(10,2);

   DECLARE cur CURSOR FOR

      SELECT product_id, stock, price FROM products;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

   OPEN cur;

   read_loop: LOOP

      FETCH cur INTO pid, pstock, pprice;

      IF done THEN

         LEAVE read_loop;

      END IF;
```

```
        IF pstock < 5 THEN

            UPDATE products

            SET price = pprice + (pprice * 0.10)

            WHERE product_id = pid;

        END IF;

    END LOOP;

    CLOSE cur;

END //

DELIMITER ;

CALL increase_price_low_stock();

SELECT product_id, product_name, price, stock FROM products;
```

Output:-



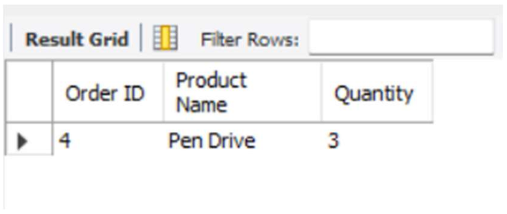
product_id	product_name	price	stock
1	Laptop	55000.00	8
2	Mouse	500.00	20
3	Keyboard	1500.00	5
4	Monitor	13200.00	1
5	Pen Drive	880.00	-3
NULL	NULL	NULL	NULL

Question 10: Display all orders along with their corresponding product names and quantities using a cursor.

Code:-

```
DELIMITER //

CREATE PROCEDURE show_order_details()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE o_id INT;
    DECLARE p_name VARCHAR(100);
    DECLARE qty INT;

    DECLARE cur CURSOR FOR
        SELECT o.order_id, p.product_name, o.quantity
        FROM orders o
        JOIN products p ON o.product_id = p.product_id;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO o_id, p_name, qty;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT o_id AS 'Order ID', p_name AS 'Product Name', qty AS 'Quantity';
    END LOOP;

    CLOSE cur;
END //

DELIMITER ;

CALL show_order_details();
```

Output:-



Order ID	Product Name	Quantity
4	Pen Drive	3