Assignment - 6

DBMS

Name: Kamithkar Vinod
Course: PG DAC AUGUST 2025
PRN: 250850320040
Form No: 250500480
Date: 28-10-2025

Problem 1: CURSOR

Task: Update the total$_amountintheorderstableusingthecorrespondingproductpricefromtheproductst$

Code:

```
DELIMITER //
CREATE PROCEDURE update_orders_total_amount()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE oid INT;
    DECLARE pid INT;
    DECLARE qty INT;
    DECLARE pr DECIMAL(10,2);
    DECLARE cur CURSOR FOR SELECT order_id, product_id, quantity
       FROM orders;
    DECLARE EXIT HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    cur_loop: LOOP
        FETCH cur INTO oid, pid, qty;
        IF done THEN
            LEAVE cur_loop;
        END IF;
        SELECT price INTO pr FROM products WHERE product_id = pid
           ;
        UPDATE orders SET total_amount = pr * qty WHERE order_id
           = oid;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
```

```
CALL update_orders_total_amount();
```

Output: —

```
mysql> select * from orders;
+----------+------------+----------+--------------+------------+
| order_id | product_id | quantity | total_amount | order_date |
+----------+------------+----------+--------------+------------+
|        1 |          1 |        1 |     55000.00 | 2025-10-01 |
|        2 |          2 |        5 |      3000.00 | 2025-10-02 |
|        3 |          3 |        3 |      3600.00 | 2025-10-02 |
|        4 |          4 |        2 |     17000.00 | 2025-10-03 |
|        5 |          5 |       10 |      2500.00 | 2025-10-03 |
|        6 |          6 |        1 |      9500.00 | 2025-10-04 |
|        7 |          7 |        4 |      7200.00 | 2025-10-04 |
|        8 |          8 |        2 |      9600.00 | 2025-10-05 |
|        9 |          9 |        2 |      4200.00 | 2025-10-06 |
|       10 |         10 |        3 |      9600.00 | 2025-10-06 |
+----------+------------+----------+--------------+------------+
```

Problem 2: CURSOR

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

Code: —

```
DELIMITER //
CREATE PROCEDURE fetch_product_names()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pname VARCHAR(50);
    DECLARE cur CURSOR FOR SELECT product_name FROM products;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    CREATE TEMPORARY TABLE names(name VARCHAR(50));

    OPEN cur;
    cur_loop: LOOP
        FETCH cur INTO pname;
        IF done THEN
            LEAVE cur_loop;
        END IF;
        INSERT INTO names VALUES (pname);
    END LOOP;
    CLOSE cur;

    SELECT * FROM names;
    DROP TEMPORARY TABLE IF EXISTS names;
END //
DELIMITER ;

CALL fetch_product_names();
```

Output: —

```
mysql> CALL fetch_product_names();
+--------------------+
| name               |
+--------------------+
| Laptop             |
| Mouse              |
| Keyboard           |
| Monitor            |
| USB Cable          |
| Printer            |
| Headphones         |
| External Hard Disk |
| Webcam             |
| Speakers           |
+--------------------+
```

Problem 3: CURSOR

Task: Copy all data from the orders table into the order$_audittableusingacursor$.

Code: —

```
DELIMITER //
CREATE PROCEDURE copy_orders_to_audit()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE oid INT;
    DECLARE pid INT;
    DECLARE tamt DECIMAL(10,2);
    DECLARE cur CURSOR FOR SELECT order_id, product_id,
       total_amount FROM orders;
        DECLARE EXIT HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    loop1: LOOP
        FETCH cur INTO oid, pid, tamt;
        IF done THEN
            LEAVE loop1;
        END IF;
        INSERT INTO order_audit VALUES (oid, pid, tamt);
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL copy_orders_to_audit();

select * from order_audit;
```

Output: —

```
mysql> select * from order_audit;
+----------+------------+--------------+
| order_id | product_id | total_amount |
+----------+------------+--------------+
|        1 |          1 |     55000.00 |
|        2 |          2 |      3000.00 |
|        3 |          3 |      3600.00 |
|        4 |          4 |     17000.00 |
|        5 |          5 |      2500.00 |
|        6 |          6 |      9500.00 |
|        7 |          7 |      7200.00 |
|        8 |          8 |      9600.00 |
|        9 |          9 |      4200.00 |
|       10 |         10 |      9600.00 |
+----------+------------+--------------+
```

Problem 4: CURSOR

Task: Reduce the stock in the products table for each order processed according to the ordered quantity.

Code: —

```
DELIMITER //
CREATE PROCEDURE reduce_stock_after_orders()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pid INT;
    DECLARE qty INT;
    DECLARE cur CURSOR FOR SELECT product_id, quantity FROM
       orders;
    DECLARE EXIT HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    stock_loop: LOOP
        FETCH cur INTO pid, qty;
        IF done THEN
            LEAVE stock_loop;
        END IF;
        UPDATE products SET stock = stock - qty WHERE product_id
           = pid;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL reduce_stock_after_orders();
```

Output: —

```
mysql> select * from products;
+------------+--------------------+----------+-------+
| product_id | product_name       | price    | stock |
+------------+--------------------+----------+-------+
|          1 | Laptop             | 55000.00 |     4 |
|          2 | Mouse              |   600.00 |    40 |
|          3 | Keyboard           |  1200.00 |    27 |
|          4 | Monitor            |  8500.00 |     6 |
|          5 | USB Cable          |   250.00 |    90 |
|          6 | Printer            |  9500.00 |     3 |
|          7 | Headphones         |  1800.00 |    16 |
|          8 | External Hard Disk |  4800.00 |     4 |
|          9 | Webcam             |  2100.00 |     7 |
|         10 | Speakers           |  3200.00 |    12 |
+------------+--------------------+----------+-------+
```

Problem 5: CURSOR

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

Code: —

```
DELIMITER //
CREATE PROCEDURE display_out_of_stock()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pname VARCHAR(50);
    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 CONCAT(pname, ' is out of stock') AS message;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL display_out_of_stock();
```

Output: —

```
mysql> CALL display_out_of_stock();
Query OK, 0 rows affected (0.00 sec)
```

Problem 6: CURSOR

Task: 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 pr DECIMAL(10,2);
    DECLARE total DECIMAL(10,2) DEFAULT 0;
    DECLARE countp INT DEFAULT 0;
    DECLARE cur CURSOR FOR SELECT price FROM products;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    loop_prices: LOOP
        FETCH cur INTO pr;
        IF done THEN
            LEAVE loop_prices;
        END IF;
        SET total = total + pr;
        SET countp = countp + 1;
    END LOOP;
    CLOSE cur;

    SELECT total / countp AS 'Average Product Price';
END //
DELIMITER ;

CALL avg_product_price();
```

Output:

```
mysql> CALL avg_product_price();
+-----------------------+
| Average Product Price |
+-----------------------+
|           8695.000000 |
+-----------------------+
```

Problem 7: CURSOR

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

Code:

```
DELIMITER //
CREATE PROCEDURE high_value_orders()
BEGIN
    DECLARE done INT DEFAULT 0;
```

```
    DECLARE oid INT;
    DECLARE amt DECIMAL(10,2);
    DECLARE cur CURSOR FOR SELECT order_id, total_amount FROM
       orders WHERE total_amount > 10000;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    hv_loop: LOOP
        FETCH cur INTO oid, amt;
        IF done THEN
            LEAVE hv_loop;
        END IF;
        SELECT CONCAT('Order ', oid, ' amount    ', amt, ' is a
           high-value order') AS message;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL high_value_orders();
```

Output: —

```
mysql> CALL high_value_orders();
+-----------------------------------------------+
| message                                       |
+-----------------------------------------------+
| Order 1 amount ?55000.00 is a high-value order |
+-----------------------------------------------+
1 row in set (0.01 sec)

+-----------------------------------------------+
| message                                       |
+-----------------------------------------------+
| Order 4 amount ?17000.00 is a high-value order |
+-----------------------------------------------+
```

Problem 8: CURSOR

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

Code: —

```
CREATE TABLE IF NOT EXISTS product_sales_summary (
    product_id INT,
    total_quantity INT
);

DELIMITER //
CREATE PROCEDURE create_sales_summary()
```

```
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pid INT;
    DECLARE qty INT;
    DECLARE cur CURSOR FOR SELECT product_id, quantity FROM
       orders;
    DECLARE EXIT HANDLER FOR NOT FOUND SET done = 1;

    TRUNCATE TABLE product_sales_summary;

    OPEN cur;
    sum_loop: LOOP
        FETCH cur INTO pid, qty;
        IF done THEN
            LEAVE sum_loop;
        END IF;
        INSERT INTO product_sales_summary(product_id,
           total_quantity)
        VALUES (pid, qty)
        ON DUPLICATE KEY UPDATE total_quantity = total_quantity +
            qty;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL create_sales_summary();
SELECT * FROM product_sales_summary;
```

Output: —

```
mysql> SELECT * FROM product_sales_summary;
+------------+----------------+
| product_id | total_quantity |
+------------+----------------+
|          1 |              1 |
|          2 |              5 |
|          3 |              3 |
|          4 |              2 |
|          5 |             10 |
|          6 |              1 |
|          7 |              4 |
|          8 |              2 |
|          9 |              2 |
|         10 |              3 |
+------------+----------------+
```

Problem 9: CURSOR

Task: Increase the price of all products with stock less than 5 by 10

Code:

```
DELIMITER //
CREATE PROCEDURE increase_low_stock_price()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE pid INT;
    DECLARE pr DECIMAL(10,2);
    DECLARE cur CURSOR FOR SELECT product_id, price FROM products
        WHERE stock < 5;
    DECLARE EXIT HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;
    lp_loop: LOOP
        FETCH cur INTO pid, pr;
        IF done THEN
            LEAVE lp_loop;
        END IF;
        UPDATE products SET price = pr * 1.10 WHERE product_id =
           pid;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL increase_low_stock_price();
```

Output:

```
mysql> select * from products;
+------------+--------------------+----------+-------+
| product_id | product_name       | price    | stock |
+------------+--------------------+----------+-------+
|          1 | Laptop             | 60500.00 |     4 |
|          2 | Mouse              |   600.00 |    40 |
|          3 | Keyboard           |  1200.00 |    27 |
|          4 | Monitor            |  8500.00 |     6 |
|          5 | USB Cable          |   250.00 |    90 |
|          6 | Printer            | 10450.00 |     3 |
|          7 | Headphones         |  1800.00 |    16 |
|          8 | External Hard Disk |  5280.00 |     4 |
|          9 | Webcam             |  2100.00 |     7 |
|         10 | Speakers           |  3200.00 |    12 |
+------------+--------------------+----------+-------+
```

Problem 10: CURSOR

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

Code: —

```
DELIMITER //
CREATE PROCEDURE show_orders_with_product_names()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE oid INT;
    DECLARE pname VARCHAR(50);
    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 oid, pname, qty;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT CONCAT('Order ', oid, ': ', qty, ' x ', pname) AS
           details;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;

CALL show_orders_with_product_names();
```

Output: —

```
+--------------------+
| details            |
+--------------------+
| Order 9: 2 x Webcam |
+--------------------+
1 row in set (0.03 sec)

+-----------------------+
| details               |
+-----------------------+
| Order 10: 3 x Speakers |
+-----------------------+
```