Problem Statement 15 (Cursors)
Consider the following schema for Products table.
Products(Product_id, Product_Name, Product_Type, Price)
1. Write a parameterized cursor to display all products in the given price range of price and type ‘Apparel’.
Hint: Take the user input for minimum and maximum price for price range.
2. Write an explicit cursor to display information of all products with Price greater than 5000.
3. Write an implicit cursor to display the number of records affected by the update operation incrementing Price
of all products by 1000.

```
CREATE DATABASE Cursur;

use Curser;

CREATE TABLE Products (
    Product_id INT PRIMARY KEY,
    Product_Name VARCHAR(50),
    Product_Type VARCHAR(50),
    Price DECIMAL(10, 2)
);
INSERT INTO Products (Product_id, Product_Name, Product_Type, Price) VALUES
(1, 'T-Shirt', 'Apparel', 1500),
(2, 'Jeans', 'Apparel', 3000),
(3, 'Laptop', 'Electronics', 60000),
(4, 'Watch', 'Accessories', 8000),
(5, 'Jacket', 'Apparel', 4000);
```

1. Write a parameterized cursor to display all products in the given price range of price and type ‘Apparel’. Hint: Take the user input for minimum and maximum price for price range.

```
DELIMITER //

CREATE PROCEDURE GetProductsInPriceRange(min_price DECIMAL(10, 2), max_price
DECIMAL(10, 2))
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE prod_id INT;
    DECLARE prod_name VARCHAR(50);
    DECLARE prod_type VARCHAR(50);
    DECLARE prod_price DECIMAL(10, 2);

    -- Define the cursor with parameters
    DECLARE product_cursor CURSOR FOR
        SELECT Product_id, Product_Name, Product_Type, Price
        FROM Products
        WHERE Product_Type = 'Apparel' AND Price BETWEEN min_price AND max_price;

    -- Declare NOT FOUND handler
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the cursor
    OPEN product_cursor;

    -- Fetch each row and display
    product_loop: LOOP
```

```
        FETCH product_cursor INTO prod_id, prod_name, prod_type, prod_price;
        IF done THEN
            LEAVE product_loop;
        END IF;
        -- Display each record (simulating output)
        SELECT prod_id AS Product_ID, prod_name AS Product_Name, prod_type AS
Product_Type, prod_price AS Price;
    END LOOP;

    -- Close the cursor
    CLOSE product_cursor;
END//
DELIMITER ;



CALL GetProductsInPriceRange(1000, 4000);
```

2. Write an explicit cursor to display information of all products with Price greater than 5000.

```
DELIMITER //
CREATE PROCEDURE DisplayProductsAbove5000()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE prod_id INT;
    DECLARE prod_name VARCHAR(50);
    DECLARE prod_type VARCHAR(50);
    DECLARE prod_price DECIMAL(10, 2);

    -- Define the cursor
    DECLARE product_cursor CURSOR FOR
        SELECT Product_id, Product_Name, Product_Type, Price
        FROM Products
        WHERE Price > 5000;

    -- Declare NOT FOUND handler
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- Open the cursor
    OPEN product_cursor;

    -- Fetch each row and display
    product_loop: LOOP
        FETCH product_cursor INTO prod_id, prod_name, prod_type, prod_price;
        IF done THEN
            LEAVE product_loop;
        END IF;
        -- Display each record (simulating output)
        SELECT prod_id AS Product_ID, prod_name AS Product_Name, prod_type AS
Product_Type, prod_price AS Price;
    END LOOP;

    -- Close the cursor
    CLOSE product_cursor;
```

```
END//
DELIMITER ;


CALL DisplayProductsAbove5000();
```

3.Write an implicit cursor to display the number of records affected by the update operation incrementing Price of all products by 1000.

```
DELIMITER //
CREATE PROCEDURE UpdateProductPrices()
BEGIN
    -- Update statement
    UPDATE Products
    SET Price = Price + 1000;

    -- Display the number of affected rows
    SELECT ROW_COUNT() AS Records_Affected;
END//

DELIMITER ;

CALL UpdateProductPrices();
```