1. Set up a simple table for our example

CREATE TABLE products (

    product\_id INT PRIMARY KEY AUTO\_INCREMENT,

    product\_name VARCHAR(100),

    price DECIMAL(10, 2)

);

-- Insert some sample data

INSERT INTO products (product\_name, price) VALUES

('Laptop', 1200.00),

('Mouse', 25.00),

('Keyboard', 75.50);

-- Verify the initial data

SELECT \* FROM products;

-- -----------------------------------------------------------------------------

-- 2. CREATE PROCEDURE with parameters and conditional logic

-- This procedure updates a product's price.

-- It uses an IF statement to check if the new price is over a certain limit.

-- If the new price is below the limit, it doesn't update and prints a message.

-- -----------------------------------------------------------------------------

DELIMITER //

CREATE PROCEDURE update\_product\_price (

    IN p\_product\_id INT,

    IN p\_new\_price DECIMAL(10, 2)

)

BEGIN

    DECLARE old\_price DECIMAL(10, 2);

    -- Get the current price of the product

    SELECT price INTO old\_price FROM products WHERE product\_id = p\_product\_id;

    -- Conditional logic: Check if the new price is above a threshold (e.g., $100)

    IF p\_new\_price > 100.00 THEN

        -- The new price is above the threshold, so we update the product.

        UPDATE products

        SET price = p\_new\_price

        WHERE product\_id = p\_product\_id;

        -- Print a success message (using a SELECT statement for this example)

        SELECT CONCAT('Price for product ID ', p\_product\_id, ' updated from ', old\_price, ' to ', p\_new\_price) AS message;

    ELSE

        -- The new price is not above the threshold, so we do nothing.

        -- Print a warning message

        SELECT CONCAT('Price for product ID ', p\_product\_id, ' was NOT updated. New price (', p\_new\_price, ') is too low.') AS message;

    END IF;

END //

DELIMITER ;

-- -----------------------------------------------------------------------------

-- 3. CREATE FUNCTION with parameters

-- This function calculates a final price after a 5% tax is applied.

-- It returns a single DECIMAL value.

-- -----------------------------------------------------------------------------

DELIMITER //

CREATE FUNCTION calculate\_final\_price (

    p\_price DECIMAL(10, 2)

)

RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

    DECLARE final\_price DECIMAL(10, 2);

    SET final\_price = p\_price \* 1.05; -- Apply a 5% tax

    RETURN final\_price;

END //

DELIMITER ;

-- -----------------------------------------------------------------------------

-- 4. How to call the procedure and the function

-- -----------------------------------------------------------------------------

-- Call the procedure to update the laptop's price (product\_id = 1) to a valid price.

CALL update\_product\_price(1, 1500.00);

-- Call the procedure with a price that is too low to trigger the conditional logic.

CALL update\_product\_price(2, 50.00);

-- Verify the changes in the table. Only the laptop's price should have changed.

SELECT \* FROM products;

-- Use the function to calculate the final price of the keyboard (product\_id = 3).

-- Note how the function is used directly within a SELECT statement.

SELECT

    product\_name,

    price,

    calculate\_final\_price(price) AS final\_price\_with\_tax

FROM products

WHERE product\_id = 3;