**Question 1**

A company wants to implement a customer loyalty program. They have a customers table with columns customer\_id, name, and total\_purchases. They need to update each customer's loyalty points based on their total\_purchases. A simple rule is that for every $100 spent, a customer gets 1 loyalty point.

**Task:** Write a MySQL stored procedure that uses a loop to iterate through each customer and calculate their loyalty points. The procedure should update a new column, loyalty\_points, in the customers table.

**Question 2**

**Scenario:** A company wants to give a 5% raise to a specific number of its employees, starting with the employee who has the lowest salary, and continuing until a certain number of employees have been updated.

**Task:** Create a stored procedure that uses a **WHILE loop** and a variable to update the salaries of the 3 lowest-paid employees. The procedure should:

1. Declare a variable to act as a counter, initialized to 0.
2. Use a WHILE loop that continues as long as the counter is less than 3.
3. Inside the loop, update the salary of the employee with the lowest current salary by 5%.
4. Increment the counter after each update.

**Question 3**

A company manages its product inventory with a products table and an orders table. When a new order is placed, the quantity of the product ordered must be automatically deducted from the products table.

**Task:** Create an **AFTER INSERT** trigger on the orders table. This trigger should automatically update the stock\_quantity in the products table, subtracting the quantity\_ordered from the products table for the corresponding product\_id.

**Question 4**

An application allows users to enter a product's price, but the price must always be a non-negative number. To enforce this rule at the database level, a trigger is needed to prevent invalid data from being inserted.

**Task:** Create a **BEFORE INSERT** trigger on the products table. This trigger should check if the price of a new product is less than 0. If it is, the trigger should set the price to 0 and issue a warning.

**Question 5**

Create a function named calculate\_area that calculates the area of a rectangle. The function should accept two decimal numbers, length and width, as input and return their product.

**Question 6**

Create a function named is\_even that checks if an integer is even. The function should take one integer, num, as input and return the string 'Even' if the number is even, and 'Odd' if it is odd.

Self-placed

**Inventory Reorder Status**

**Scenario:** A company's inventory system needs a quick way to determine if a product is low on stock and needs to be reordered. The reorder threshold is 50 units.

**Task:** Create a MySQL **stored function** named GetReorderStatus that takes a product's current\_stock and a reorder\_threshold as input. The function should return the string 'REORDER' if the current stock is less than the threshold, and 'OK' otherwise.

-- Step 1: Create a sample table

CREATE TABLE products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(50),

stock\_quantity INT

);

-- Step 2: Insert sample data

INSERT INTO products (product\_id, product\_name, stock\_quantity) VALUES

(101, 'Smartphone', 25),

(102, 'Tablet', 75),

(103, 'Headphones', 50);

-- Step 3: Create the function

DELIMITER //

CREATE FUNCTION GetReorderStatus(current\_stock INT, reorder\_threshold INT)

RETURNS VARCHAR(10)

DETERMINISTIC

BEGIN

DECLARE status\_text VARCHAR(10);

IF current\_stock <= reorder\_threshold THEN

SET status\_text = 'REORDER';

ELSE

SET status\_text = 'OK';

END IF;

RETURN status\_text;

END//

DELIMITER ;

-- Step 4: Test the function

SELECT

product\_name,

stock\_quantity,

GetReorderStatus(stock\_quantity, 50) AS reorder\_status

FROM products;

**Employee Bonus Calculation**

**Scenario:** A company calculates employee bonuses based on their years of service and salary. The bonus is a percentage of their salary, where the percentage is equal to their years of service, up to a maximum of 15%. This calculation needs to be easily reusable across different reports and queries.

**Task:** Create a MySQL **stored function** named CalculateBonus that takes an employee's salary and years\_of\_service as input and returns the calculated bonus amount.

-- Step 1: Create a sample table

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

name VARCHAR(50),

salary DECIMAL(10, 2),

hire\_date DATE

);

-- Step 2: Insert sample data

INSERT INTO employees (employee\_id, name, salary, hire\_date) VALUES

(1, 'Alice', 60000.00, '2020-01-15'),

(2, 'Bob', 80000.00, '2010-05-20'),

(3, 'Charlie', 45000.00, '2023-08-01');

-- Step 3: Create the function

DELIMITER //

CREATE FUNCTION CalculateBonus(emp\_salary DECIMAL(10, 2), years\_of\_service INT)

RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

DECLARE bonus\_rate DECIMAL(5, 2);

DECLARE bonus\_amount DECIMAL(10, 2);

-- Set the bonus rate (max 15%)

IF years\_of\_service > 15 THEN

SET bonus\_rate = 0.15;

ELSE

SET bonus\_rate = years\_of\_service / 100;

END IF;

-- Calculate the bonus amount

SET bonus\_amount = emp\_salary \* bonus\_rate;

RETURN bonus\_amount;

END//

DELIMITER ;

-- Step 4: Test the function

SELECT

name,

salary,

TIMESTAMPDIFF(YEAR, hire\_date, CURDATE()) AS years\_of\_service,

CalculateBonus(salary, TIMESTAMPDIFF(YEAR, hire\_date, CURDATE())) AS bonus

FROM employees;