PL/SQL HANDSON

I) CONTROL STRUCTURES

Scenario 1: Apply 1% Discount for Customers Above 60

```
DELIMITER $$
CREATE PROCEDURE ApplyDiscount()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_cid INT;
    DECLARE v_age INT;
    DECLARE v_cur_loan_interest DECIMAL(5, 2);
    DECLARE customer_cursor CURSOR FOR
        SELECT cid, age, cur_loan_interest FROM customers;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN customer_cursor;
    read_loop: LOOP
        FETCH customer_cursor INTO v_cid, v_age, v_cur_loan_interest;
        IF done THEN
            LEAVE read_loop;
        END IF;
        IF v_age > 60 THEN
            UPDATE customers
            SET cur_loan_interest = v_cur_loan_interest * 0.99
            WHERE cid = v_cid;
        END IF;
    END LOOP;
    CLOSE customer_cursor;
END$$
DELIMITER ;
```

SELECT * FROM CUSTOMERS;

	cid	cname	age	cur_loan_interest	balance	is_vip
▶	1	Alice	65	7.50	12000.00	N
	2	Bob	58	6.80	8000.00	N
	3	Charlie	70	8.20	15000.00	N
	4	Diana	45	5.90	4000.00	N
	5	Edward	62	6.70	11000.00	N

Scenario 2: Set VIP Status Based on Balance

```
SET SQL_SAFE_UPDATES = 0;
UPDATE customers
SET is_vip = CASE
    WHEN balance > 10000 THEN 'Y'
    ELSE 'N'
END;

SET SQL_SAFE_UPDATES = 1;
```

SELECT * FROM CUSTOMERS;

cid	cname	age	cur_loan_interest	balance	is_vip
1	Alice	65	7.50	12000.00	Y
2	Bob	58	6.80	8000.00	N
3	Charlie	70	8.20	15000.00	Y
4	Diana	45	5.90	4000.00	N
5	Edward	62	6.70	11000.00	Y

Scenario 3: Send Reminders for Loans Due Within the Next 30 Days

```
DELIMITER $$
DROP PROCEDURE IF EXISTS SendLoanReminders$$
CREATE PROCEDURE SendLoanReminders()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE v_cid INT;
    DECLARE v_cname VARCHAR(20);
    DECLARE v_due_date DATE;
    DECLARE customer_cursor CURSOR FOR
        SELECT cid, cname, loan_due_date
        FROM customers
        WHERE loan_due_date <= CURDATE() + INTERVAL 30 DAY;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN customer_cursor;
    read_loop: LOOP
        FETCH customer_cursor INTO v_cid, v_cname, v_due_date;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT CONCAT('Reminder: Loan due for ', v_cname, ' on ', v_due_date) AS Reminder_Messa
    END LOOP;
    CLOSE customer_cursor;
END$$
DELIMITER ;
```

SELECT * FROM CUSTOMERS;

cid	cname	age	cur_loan_interest	balance	is_vip	loan_due_date
1	Alice	65	7.50	12000.00	Y	2025-07-23
2	Bob	58	6.80	8000.00	N	2025-08-07
3	Charlie	70	8.20	15000.00	Y	2025-07-08
4	Diana	45	5.90	4000.00	N	2025-07-03
5	Edward	62	6.70	11000.00	Y	2025-08-02

I) STORED PROCEDURES

<u>Scenario 1:</u> The bank needs to process monthly interest for all savings accounts.

```
-- Scenario1: Process Monthly Interest for Savings Accounts
DELIMITER $$
CREATE PROCEDURE ProcessMonthlyInterest()
BEGIN
    -- Apply a 1% interest to all balances
    UPDATE customers
    SET balance = balance + (balance * 0.01)
    WHERE balance IS NOT NULL;
END$$

DELIMITER ;

CALL ProcessMonthlyInterest();
SELECT * FROM customers;
```

Output:

cid	cname	age	cur_loan_interest	balance	is_vip	loan_due_date
1	Alice	65	7.50	12000.00	Y	2025-07-23
2	Bob	58	6.80	8000.00	N	2025-08-07
3	Charlie	70	8.20	15000.00	Y	2025-07-08
4	Diana	45	5.90	4000.00	N	2025-07-03
5	Edward	62	6.70	11000.00	Y	2025-08-02

Scenario 2: ***The bank wants to implement a bonus scheme for employees based on their performance.***

```
DELIMITER $$
CREATE PROCEDURE UpdateEmployeeBonus(
    IN dept_name VARCHAR(50),
    IN bonus_percent DECIMAL(5, 2)
)
BEGIN
    UPDATE employees
    SET bonus = bonus + (bonus * (bonus_percent / 100))
    WHERE department = dept_name;
END$$
DELIMITER ;
```

CALL UpdateEmployeeBonus('Sales', 10);
SELECT * FROM employees;

Output:

	emp_id	emp_name	department	bonus
▶	1	John	Sales	1100.00
	2	Jane	HR	800.00
	3	Alex	Sales	1320.00
	4	Sara	IT	900.00
	5	Chris	HR	1100.00
*	NULL	NULL	NULL	NULL

<u>Scenario 3:</u> Customers should be able to transfer funds between their accounts.

```
DELIMITER $$
CREATE PROCEDURE TransferFunds(
    IN source_account INT,
    IN target_account INT,
    IN transfer_amount DECIMAL(10, 2)
)
BEGIN
    DECLARE insufficient_balance INT DEFAULT 0;
    SELECT CASE WHEN balance < transfer_amount THEN 1 ELSE 0 END
    INTO insufficient_balance
    FROM accounts
    WHERE account_id = source_account;
    IF insufficient_balance = 1 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Insufficient balance in the source account';
    ELSE
        UPDATE accounts
        SET balance = balance - transfer_amount
        WHERE account_id = source_account;
        UPDATE accounts
        SET balance = balance + transfer_amount
        WHERE account_id = target_account;
    END IF;
END$$
DELIMITER ;
CALL TransferFunds(1, 2, 500);
```

SELECT * FROM accounts;

Output:

account_id	customer_id	balance
1	101	4500.00
2	102	3500.00
3	103	7000.00
4	104	4000.00
5	105	2000.00