**Candidate Name** :-  MEGHANA G K

**Superset ID** :- 6424295

**WEEK – 2 HANDS ON EXERCISE (JAVA FSE DEEPSKILLING)**

**(PL/SQL PROGRAMMING)**

**Exercise 1: Control Structures**

**Scenario 1 :**

     The bank wants to apply a discount to loan interest rates for customers above 60 yearsold.

**Question :** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Code for the above question:-**

CREATE TABLE loan\_customers (

    customer\_id NUMBER,

    interest\_rate NUMBER,

    age NUMBER

);

INSERT INTO loan\_customers VALUES (101, 10.5, 65);

INSERT INTO loan\_customers VALUES (102, 12.0, 55);

INSERT INTO loan\_customers VALUES (103, 11.5, 70);

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

    CURSOR customer\_cursor IS

        SELECT customer\_id, interest\_rate, age FROM loan\_customer

   new\_rate NUMBER;

BEGIN

    FOR rec IN customer\_cursor LOOP

        IF rec.age > 60 THEN

            new\_rate := rec.interest\_rate - 1;

            UPDATE loan\_customers

            SET interest\_rate = new\_rate

            WHERE customer\_id = rec.customer\_id;

          DBMS\_OUTPUT.PUT\_LINE('Interest rate updated for senior customer ID: ' ||  rec.customer\_id);

        END IF;

    END LOOP;

END;

**Output Image:-**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2 :**

A customer can be promoted to VIP status based on their balance.

**Question :**  Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000..

**Code for the above question:-**

CREATE TABLE account\_holders (

    customer\_id NUMBER,

    balance NUMBER,

    isvip VARCHAR2(10)

);

INSERT INTO account\_holders VALUES (201, 9500, 'FALSE');

INSERT INTO account\_holders VALUES (202, 12000, 'FALSE');

INSERT INTO account\_holders VALUES (203, 15000, 'FALSE');

INSERT INTO account\_holders VALUES (204, 8000, 'FALSE');

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

    CURSOR balance\_cursor IS

        SELECT customer\_id, balance FROM account\_holders;

BEGIN

    FOR rec IN balance\_cursor LOOP

        IF rec.balance > 10000 THEN

            UPDATE account\_holders

            SET isvip = 'TRUE'

            WHERE customer\_id = rec.customer\_id;

            DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || rec.customer\_id || ' marked as VIP.');

        END IF;

    END LOOP;

END;

**Output Image:-**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 3 :**

The bank wants to send reminders to customers whose loans are due within the next 30 days**.**

**Question :**  Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Code for the above question:-**

CREATE TABLE loan\_accounts (

    loan\_id NUMBER,

   customer\_id NUMBER,

    due\_date DATE

);

INSERT INTO loan\_accounts VALUES (501, 301, TO\_DATE('10-JUL-2025', 'DD-MON-YYYY'));

INSERT INTO loan\_accounts VALUES (502, 302, TO\_DATE('20-JUN-2025', 'DD-MON-YYYY'));

INSERT INTO loan\_accounts VALUES (503, 303, TO\_DATE('05-JUL-2025', 'DD-MON-YYYY'));

INSERT INTO loan\_accounts VALUES (504, 304, TO\_DATE('30-JUL-2025', 'DD-MON-YYYY'));

COMMIT;

SET SERVEROUTPUT ON;

DECLARE

    CURSOR upcoming\_loans IS

        SELECT loan\_id, customer\_id, due\_date

        FROM loan\_accounts

        WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

    FOR record IN upcoming\_loans LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ' || record.customer\_id ||

                             ' has a loan (ID: ' || record.loan\_id ||

                             ') due on ' || TO\_CHAR(record.due\_date, 'DD-MON-YYYY'));

    END LOOP;

END;

**Output Image:-**

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 3: Stored Procedures**

**Scenario 1 :**

      The bank needs to process monthly interest for all savings accounts.

**Question :**  Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Code for above question:-**

CREATE TABLE savings\_accounts (

    account\_number NUMBER PRIMARY KEY,

    account\_holder VARCHAR2(50),

    balance NUMBER

);

INSERT INTO savings\_accounts VALUES (101, 'Ravi', 10000);

INSERT INTO savings\_accounts VALUES (102, 'Meena', 15000);

INSERT INTO savings\_accounts VALUES (103, 'Karan', 8000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

    UPDATE savings\_accounts

    SET balance = balance + (balance \* 0.01);

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

        ROLLBACK;

END;

/

BEGIN

    ProcessMonthlyInterest;

END;

**Output Image:-**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2 :**

   The bank wants to implement a bonus scheme for employees based on their performance.

**Question :**  Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter**.**

**Code for above question:-**

CREATE TABLE department\_employees (

    emp\_id NUMBER PRIMARY KEY,

    emp\_name VARCHAR2(50),

    department VARCHAR2(50),

    salary NUMBER

);

INSERT INTO department\_employees VALUES (201, 'Rina', 'HR', 30000);

INSERT INTO department\_employees VALUES (202, 'Ajay', 'Finance', 40000);

INSERT INTO department\_employees VALUES (203, 'Neha', 'HR', 35000);

INSERT INTO department\_employees VALUES (204, 'Vikas', 'IT', 45000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

    dept\_name IN VARCHAR2,

    bonus\_percent IN NUMBER

) IS

BEGIN

    UPDATE department\_employees

    SET salary = salary + (salary \* bonus\_percent / 100)

    WHERE department = dept\_name;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || bonus\_percent || '% applied to department: ' || dept\_name);

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Error while applying bonus: ' || SQLERRM);

        ROLLBACK;

END;

/

BEGIN

    UpdateEmployeeBonus('HR', 10);

END;

**Output Image:-**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 3 :**

     Customers should be able to transfer funds between their accounts.

**Question :**  Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**Code for above question:-**

CREATE TABLE customer\_accounts (

    acc\_id NUMBER PRIMARY KEY,

    holder\_name VARCHAR2(50),

    balance NUMBER

);

INSERT INTO customer\_accounts VALUES (1001, 'Ramesh', 6000);

INSERT INTO customer\_accounts VALUES (1002, 'Suresh', 4000);

INSERT INTO customer\_accounts VALUES (1003, 'Mahesh', 2000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

    sender\_id IN NUMBER,

    receiver\_id IN NUMBER,

    amount IN NUMBER

) IS

    sender\_balance NUMBER;

BEGIN

    SELECT balance INTO sender\_balance

    FROM customer\_accounts

    WHERE acc\_id = sender\_id;

    IF sender\_balance < amount THEN

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Not enough balance in sender account.');

        RETURN;

    END IF;

    UPDATE customer\_accounts

    SET balance = balance - amount

    WHERE acc\_id = sender\_id;

    UPDATE customer\_accounts

    SET balance = balance + amount

    WHERE acc\_id = receiver\_id;

    COMMIT;

    DBMS\_OUTPUT.PUT\_LINE(

        'Transfer of Rs. ' || amount ||

        ' from account ' || sender\_id ||

        ' to account ' || receiver\_id || ' completed successfully.'

    );

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Account not found.');

        ROLLBACK;

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

        ROLLBACK;

END;

/

BEGIN

    TransferFunds(1001, 1002, 1000);

END;

/

BEGIN

    TransferFunds(1003, 1002, 5000);

END;

/

**Output Image:-**

A screenshot of a computer

AI-generated content may be incorrect.