**Exercise 1: Control Structures**

CREATE TABLE CUSTOMERS (

    CUSTOMER\_ID NUMBER PRIMARY KEY,

    CUSTOMER\_NAME VARCHAR2(50),

    AGE NUMBER,

    BALANCE NUMBER,

    ISVIP VARCHAR2(5),

    LOAN\_INTEREST\_RATE NUMBER

);

CREATE TABLE LOANS (

    LOAN\_ID NUMBER PRIMARY KEY,

    CUSTOMER\_ID NUMBER,

    DUE\_DATE DATE,

    FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMERS(CUSTOMER\_ID)

);

INSERT INTO CUSTOMERS VALUES (1, 'Meghzzz', 62, 15000, 'FALSE', 12.5);

INSERT INTO CUSTOMERS VALUES (2, 'Aryan', 45, 8000, 'FALSE', 14.0);

INSERT INTO CUSTOMERS VALUES (3, 'Nisha', 65, 12000, 'FALSE', 13.0);

INSERT INTO LOANS VALUES (101, 1, SYSDATE + 20);

INSERT INTO LOANS VALUES (102, 2, SYSDATE + 40);

INSERT INTO LOANS VALUES (103, 3, SYSDATE + 10);

COMMIT;

BEGIN

    FOR rec IN (SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE AGE > 60) LOOP

        UPDATE CUSTOMERS

        SET LOAN\_INTEREST\_RATE = LOAN\_INTEREST\_RATE - 1

        WHERE CUSTOMER\_ID = rec.CUSTOMER\_ID;

    END LOOP;

    COMMIT;

END;

/

BEGIN

    FOR rec IN (SELECT CUSTOMER\_ID FROM CUSTOMERS WHERE BALANCE > 10000) LOOP

        UPDATE CUSTOMERS

        SET ISVIP = 'TRUE'

        WHERE CUSTOMER\_ID = rec.CUSTOMER\_ID;

    END LOOP;

    COMMIT;

END;

/

BEGIN

    FOR rec IN (

        SELECT L.LOAN\_ID, C.CUSTOMER\_NAME, L.DUE\_DATE

        FROM LOANS L

        JOIN CUSTOMERS C ON L.CUSTOMER\_ID = C.CUSTOMER\_ID

        WHERE L.DUE\_DATE BETWEEN SYSDATE AND SYSDATE + 30

    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.CUSTOMER\_NAME ||

    ', your loan (ID: ' || rec.LOAN\_ID ||') is due on ' || TO\_CHAR(rec.DUE\_DATE, 'DD-MON-YYYY'));

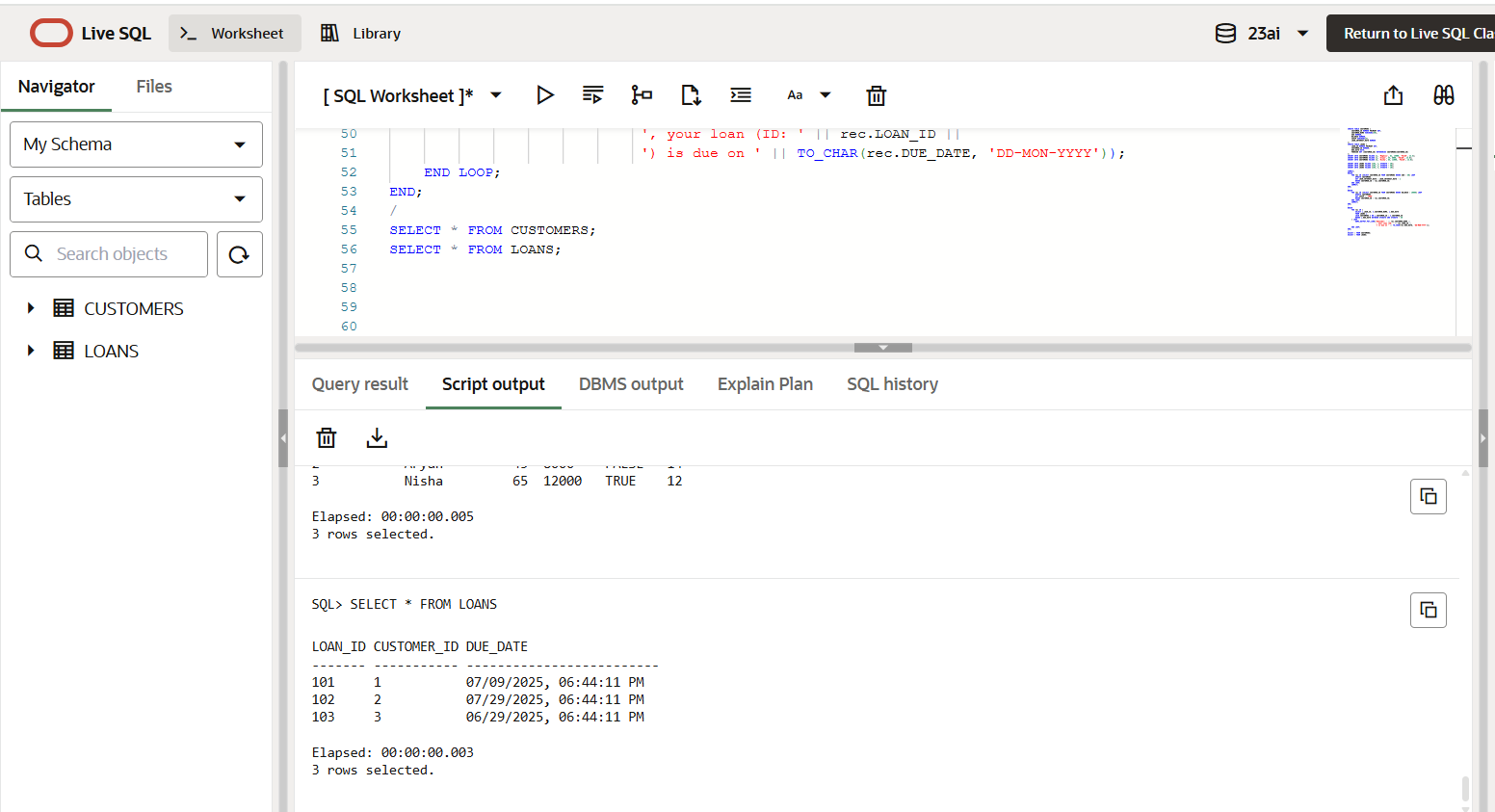
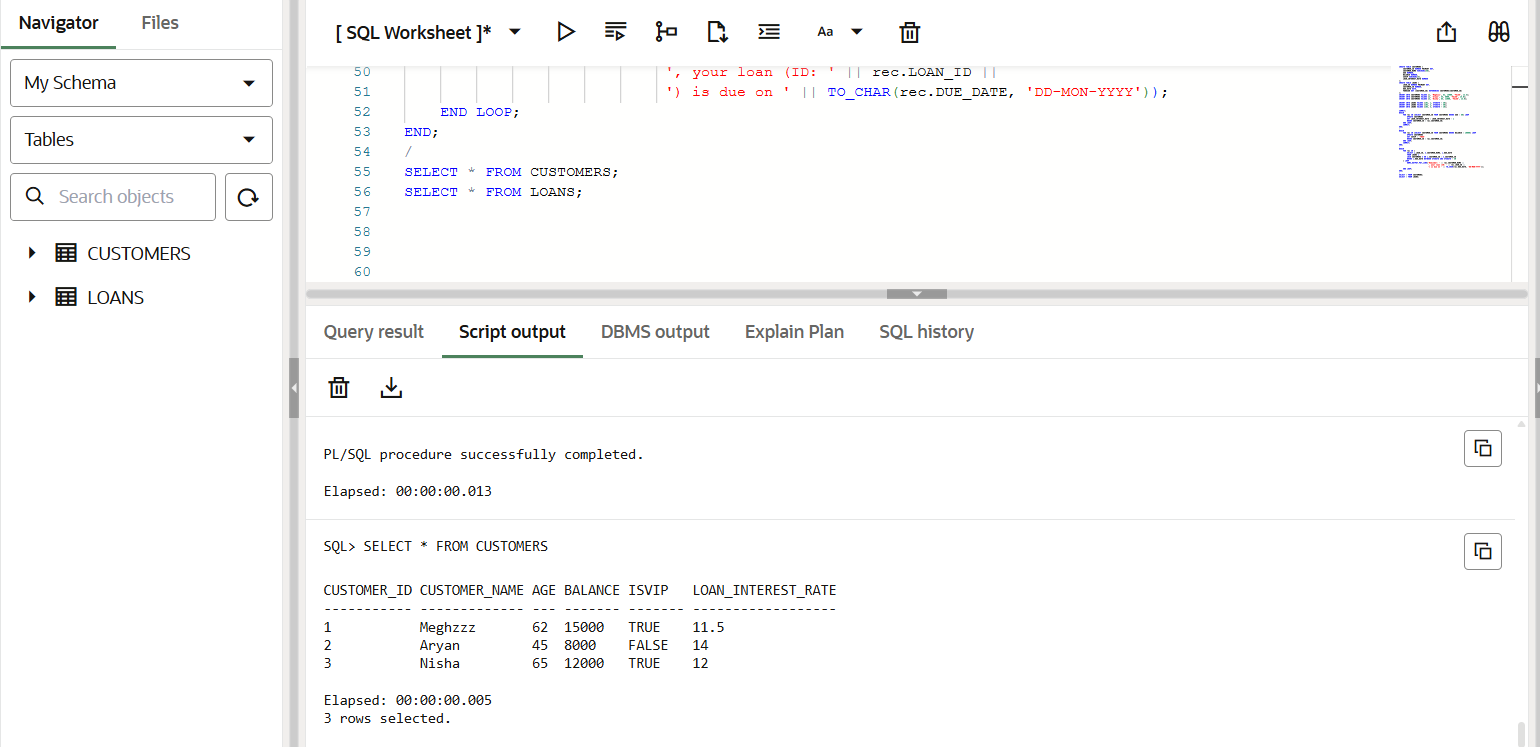
    END LOOP;

END;

/

SELECT \* FROM CUSTOMERS;

SELECT \* FROM LOANS;



**Exercise 3: Stored Procedures**

BEGIN

    EXECUTE IMMEDIATE 'DROP TABLE ACCOUNTS';

EXCEPTION

    WHEN OTHERS THEN

        NULL;

END;

/

BEGIN

    EXECUTE IMMEDIATE 'DROP TABLE EMPLOYEES';

EXCEPTION

    WHEN OTHERS THEN

        NULL;

END;

/

CREATE TABLE ACCOUNTS (

    ACCOUNT\_ID NUMBER PRIMARY KEY,

    CUSTOMER\_NAME VARCHAR2(50),

    BALANCE NUMBER,

    ACCOUNT\_TYPE VARCHAR2(20)

);

CREATE TABLE EMPLOYEES (

    EMPLOYEE\_ID NUMBER PRIMARY KEY,

    EMPLOYEE\_NAME VARCHAR2(50),

    DEPARTMENT VARCHAR2(30),

    SALARY NUMBER

);

INSERT INTO ACCOUNTS VALUES (101, 'Meghzzz', 5000, 'Savings');

INSERT INTO ACCOUNTS VALUES (102, 'Aryan', 7000, 'Savings');

INSERT INTO ACCOUNTS VALUES (103, 'Nisha', 6000, 'Current');

INSERT INTO EMPLOYEES VALUES (1, 'Karan', 'Finance', 30000);

INSERT INTO EMPLOYEES VALUES (2, 'Rhea', 'IT', 40000);

INSERT INTO EMPLOYEES VALUES (3, 'Nikki', 'Finance', 35000);

COMMIT;

-- Scenario 1: Process Monthly Interest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

    FOR acc\_rec IN (SELECT ACCOUNT\_ID FROM ACCOUNTS WHERE ACCOUNT\_TYPE = 'Savings') LOOP

        UPDATE ACCOUNTS

        SET BALANCE = BALANCE + (BALANCE \* 0.01)

        WHERE ACCOUNT\_ID = acc\_rec.ACCOUNT\_ID;

    END LOOP;

    COMMIT;

END;

/

-- Scenario 2: Update Employee Bonus by Department

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

    dept\_name IN VARCHAR2,

    bonus\_percent IN NUMBER

) AS

BEGIN

    UPDATE EMPLOYEES

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

    WHERE DEPARTMENT = dept\_name;

    COMMIT;

END;

/

-- Scenario 3: Transfer Funds Between Accounts

CREATE OR REPLACE PROCEDURE TransferFunds (

    from\_account IN NUMBER,

    to\_account IN NUMBER,

    amount IN NUMBER

) AS

    insufficient\_balance EXCEPTION;

BEGIN

    DECLARE

        source\_balance NUMBER;

    BEGIN

        SELECT BALANCE INTO source\_balance FROM ACCOUNTS WHERE ACCOUNT\_ID = from\_account;

        IF source\_balance < amount THEN

            RAISE insufficient\_balance;

        END IF;

        UPDATE ACCOUNTS

        SET BALANCE = BALANCE - amount

        WHERE ACCOUNT\_ID = from\_account;

        UPDATE ACCOUNTS

        SET BALANCE = BALANCE + amount

        WHERE ACCOUNT\_ID = to\_account;

        COMMIT;

        DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount || ' successful from ' || from\_account || ' to ' || to\_account);

    END;

EXCEPTION

    WHEN insufficient\_balance THEN

        DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

    WHEN NO\_DATA\_FOUND THEN

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

END;

/

-- Run Scenarios

BEGIN

    ProcessMonthlyInterest;

END;

/

BEGIN

    UpdateEmployeeBonus('Finance', 10);

END;

/

BEGIN

    TransferFunds(101, 102, 2000);

END;

/

SELECT \* FROM ACCOUNTS;

SELECT \* FROM EMPLOYEES;

