**PLSQL\_Exercises**

**Exercise 1: Control Structures**

**Code**

**sql**

**-- CLEANUP (optional, avoids conflict on rerun)**

**BEGIN**

**EXECUTE IMMEDIATE 'DROP TABLE LOAN';**

**EXECUTE IMMEDIATE 'DROP TABLE CUST';**

**EXCEPTION**

**WHEN OTHERS THEN NULL;**

**END;**

**/**

**-- CREATE TABLES**

**CREATE TABLE CUST (**

**CUST\_ID NUMBER PRIMARY KEY,**

**CUST\_NAME VARCHAR2(100),**

**CUST\_DOB DATE,**

**CUST\_BAL NUMBER,**

**CUST\_LAST\_MOD DATE**

**);**

**CREATE TABLE LOAN (**

**LOAN\_ID NUMBER PRIMARY KEY,**

**CUST\_ID NUMBER,**

**LOAN\_AMT NUMBER,**

**LOAN\_INT\_RATE NUMBER,**

**LOAN\_START\_DATE DATE,**

**LOAN\_END\_DATE DATE,**

**FOREIGN KEY (CUST\_ID) REFERENCES CUST(CUST\_ID)**

**);**

**-- INSERT SAMPLE DATA**

**INSERT INTO CUST VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 12000, SYSDATE);**

**INSERT INTO CUST VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 8000, SYSDATE);**

**INSERT INTO LOAN VALUES (101, 1, 5000, 5, SYSDATE, SYSDATE + 15);**

**INSERT INTO LOAN VALUES (102, 2, 7000, 6, SYSDATE, SYSDATE + 45);**

**COMMIT;**

**-- ENABLE SERVER OUTPUT**

**SET SERVEROUTPUT ON;**

**-- SCENARIO 1: Apply Interest Discount for Senior Citizens**

**BEGIN**

**FOR rec IN (**

**SELECT CUST\_ID, EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM CUST\_DOB) AS AGE**

**FROM CUST**

**) LOOP**

**IF rec.AGE > 60 THEN**

**UPDATE LOAN SET LOAN\_INT\_RATE = LOAN\_INT\_RATE - 1 WHERE CUST\_ID = rec.CUST\_ID;**

**DBMS\_OUTPUT.PUT\_LINE('Interest reduced for Customer ID: ' || rec.CUST\_ID);**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('No change for Customer ID: ' || rec.CUST\_ID || ', Age: ' || rec.AGE);**

**END IF;**

**END LOOP;**

**COMMIT;**

**END;**

**/**

**-- SCENARIO 2: Add VIP column based on balance**

**ALTER TABLE CUST ADD CUST\_VIP CHAR(5) CHECK (CUST\_VIP IN ('TRUE','FALSE'));**

**BEGIN**

**FOR rec IN (SELECT CUST\_ID, CUST\_BAL FROM CUST) LOOP**

**IF rec.CUST\_BAL > 10000 THEN**

**UPDATE CUST SET CUST\_VIP = 'TRUE' WHERE CUST\_ID = rec.CUST\_ID;**

**DBMS\_OUTPUT.PUT\_LINE('VIP Assigned to Customer ID: ' || rec.CUST\_ID);**

**ELSE**

**UPDATE CUST SET CUST\_VIP = 'FALSE' WHERE CUST\_ID = rec.CUST\_ID;**

**DBMS\_OUTPUT.PUT\_LINE('No VIP for Customer ID: ' || rec.CUST\_ID);**

**END IF;**

**END LOOP;**

**COMMIT;**

**END;**

**/**

**-- SCENARIO 3: Loan due in 30 days reminder**

**BEGIN**

**DECLARE**

**found BOOLEAN := FALSE;**

**BEGIN**

**FOR rec IN (**

**SELECT l.LOAN\_ID, l.LOAN\_END\_DATE, c.CUST\_NAME**

**FROM LOAN l**

**JOIN CUST c ON l.CUST\_ID = c.CUST\_ID**

**WHERE l.LOAN\_END\_DATE BETWEEN SYSDATE AND SYSDATE + 30**

**) LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.LOAN\_ID || ' for ' || rec.CUST\_NAME || ' is due on ' || TO\_CHAR(rec.LOAN\_END\_DATE, 'YYYY-MM-DD'));**

**found := TRUE;**

**END LOOP;**

**IF NOT found THEN**

**DBMS\_OUTPUT.PUT\_LINE('No loans due in the next 30 days.');**

**END IF;**

**END;**

**END;**

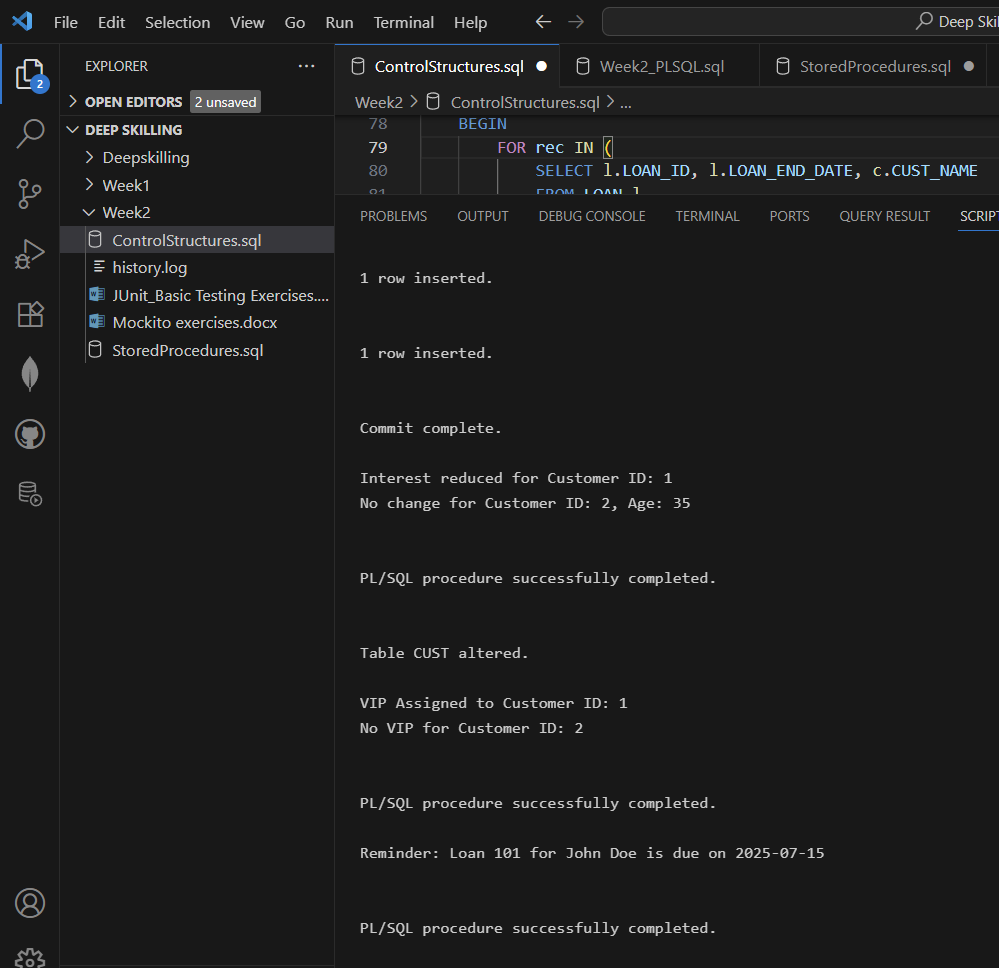
**/**

**-- VIEW FINAL DATA**

**SELECT \* FROM CUST;**

**SELECT \* FROM LOAN;**

**Output**

****

**Exercise 3: Stored Procedures**

**Code**

**SET SERVEROUTPUT ON;**

**-- Procedure 1: Apply 1% interest to savings accounts**

**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS**

**BEGIN**

**FOR acc\_rec IN (**

**SELECT AccountID, Balance**

**FROM Accounts**

**WHERE AccountType = 'Savings'**

**) LOOP**

**UPDATE Accounts**

**SET Balance = acc\_rec.Balance + (acc\_rec.Balance \* 0.01),**

**LastModified = SYSDATE**

**WHERE AccountID = acc\_rec.AccountID;**

**DBMS\_OUTPUT.PUT\_LINE('→ Interest applied to Savings Account ID: ' || acc\_rec.AccountID);**

**END LOOP;**

**END;**

**/**

**-- Procedure 2: Add bonus to employee salaries in a department**

**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(**

**dept\_name IN VARCHAR2,**

**bonus\_percent IN NUMBER**

**) AS**

**BEGIN**

**FOR emp\_rec IN (**

**SELECT EmployeeID, Salary**

**FROM Employees**

**WHERE Department = dept\_name**

**) LOOP**

**UPDATE Employees**

**SET Salary = emp\_rec.Salary + (emp\_rec.Salary \* bonus\_percent / 100)**

**WHERE EmployeeID = emp\_rec.EmployeeID;**

**DBMS\_OUTPUT.PUT\_LINE('→ Bonus of ' || bonus\_percent || '% added to Employee ID: ' || emp\_rec.EmployeeID);**

**END LOOP;**

**END;**

**/**

**-- Procedure 3: Transfer funds between two accounts**

**CREATE OR REPLACE PROCEDURE TransferFunds(**

**from\_acc IN NUMBER,**

**to\_acc IN NUMBER,**

**amount IN NUMBER**

**) AS**

**current\_balance NUMBER;**

**BEGIN**

**SELECT Balance INTO current\_balance**

**FROM Accounts**

**WHERE AccountID = from\_acc**

**FOR UPDATE;**

**IF current\_balance >= amount THEN**

**UPDATE Accounts**

**SET Balance = Balance - amount,**

**LastModified = SYSDATE**

**WHERE AccountID = from\_acc;**

**UPDATE Accounts**

**SET Balance = Balance + amount,**

**LastModified = SYSDATE**

**WHERE AccountID = to\_acc;**

**DBMS\_OUTPUT.PUT\_LINE('Transferred $' || amount || ' from Account ' || from\_acc || ' to Account ' || to\_acc);**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ' || from\_acc);**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('Error: One or both accounts not found.');**

**WHEN OTHERS THEN**

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

**END;**

**/**

**-- Execution block to run all three procedures**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('--- Running Stored Procedures ---');**

**ProcessMonthlyInterest;**

**UpdateEmployeeBonus('IT', 10);**

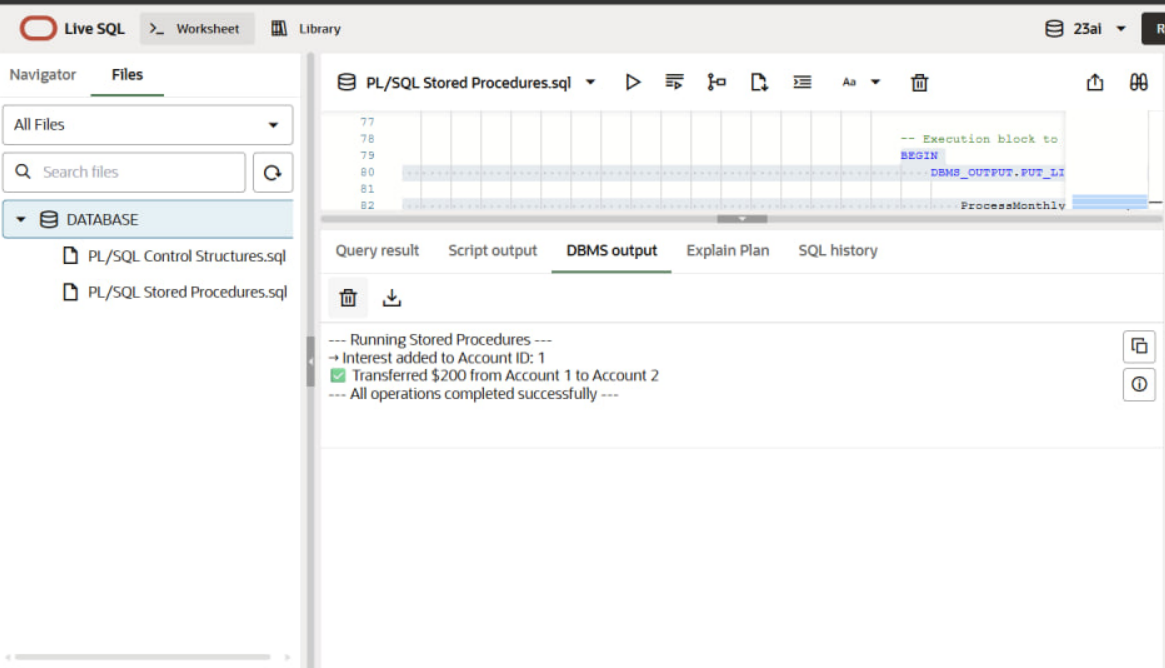
**TransferFunds(1, 2, 200);**

**DBMS\_OUTPUT.PUT\_LINE('--- All operations completed successfully ---');**

**END;**

**/**

**Output**

****