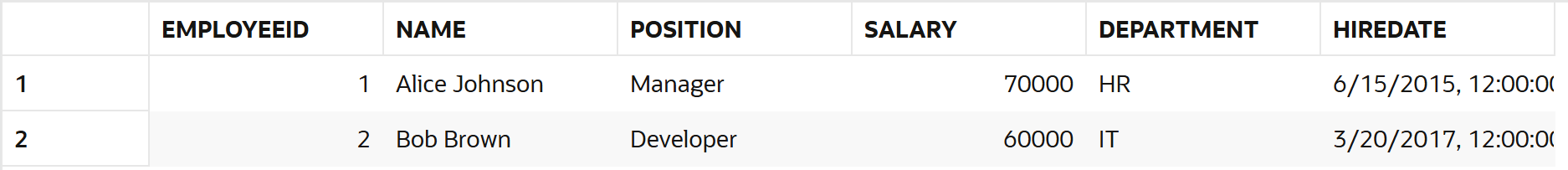
***Name: Amirtha A  
superset Id: 6394120***

***Exercise 1: Control Structures***

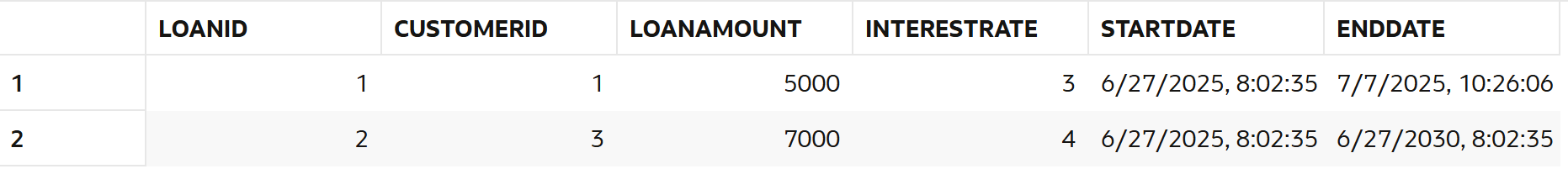
***Tables:  
  
CUSTOMERS***A screenshot of a computer

AI-generated content may be incorrect.

***EMPLOYEES***

***ACCOUNTS***A screenshot of a computer

AI-generated content may be incorrect.

***LOANS***

***TRANSACTIONS***A close up of numbers

AI-generated content may be incorrect.

***Scenario 1:*** *The bank wants to apply a discount to loan interest rates for customers above 60 years old.*

* + ***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:***

*SET SERVEROUTPUT ON;*

*DECLARE*

*cust\_id Customers.CustomerID%TYPE;*

*dob Customers.DOB%TYPE;*

*loan\_id Loans.LoanID%TYPE;*

*age NUMBER;*

*BEGIN*

*FOR rec IN (*

*SELECT c.CustomerID, c.DOB, l.LoanID*

*FROM Customers c*

*JOIN Loans l ON c.CustomerID = l.CustomerID*

*) LOOP*

*age := FLOOR(MONTHS\_BETWEEN(SYSDATE, rec.DOB) / 12);*

*IF age > 60 THEN*

*UPDATE Loans*

*SET InterestRate = InterestRate - 1*

*WHERE LoanID = rec.LoanID;*

*DBMS\_OUTPUT.PUT\_LINE('Discount given to Customer ' || rec.CustomerID);*

*END IF;*

*END LOOP;*

*COMMIT;*

*END;*

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:***

*BEGIN*

*FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP*

*IF rec.Balance > 10000 THEN*

*UPDATE Customers*

*SET IsVIP = 'YES'*

*WHERE CustomerID = rec.CustomerID;*

*END IF;*

*END LOOP;*

*COMMIT;*

*END;*

*/*

*Output:*

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:  
SET SERVEROUTPUT ON;*

*BEGIN*

*FOR rec IN (*

*SELECT c.CustomerID, c.Name, l.LoanID, l.EndDate*

*FROM Customers c*

*JOIN Loans l ON c.CustomerID = l.CustomerID*

*WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30*

*) LOOP*

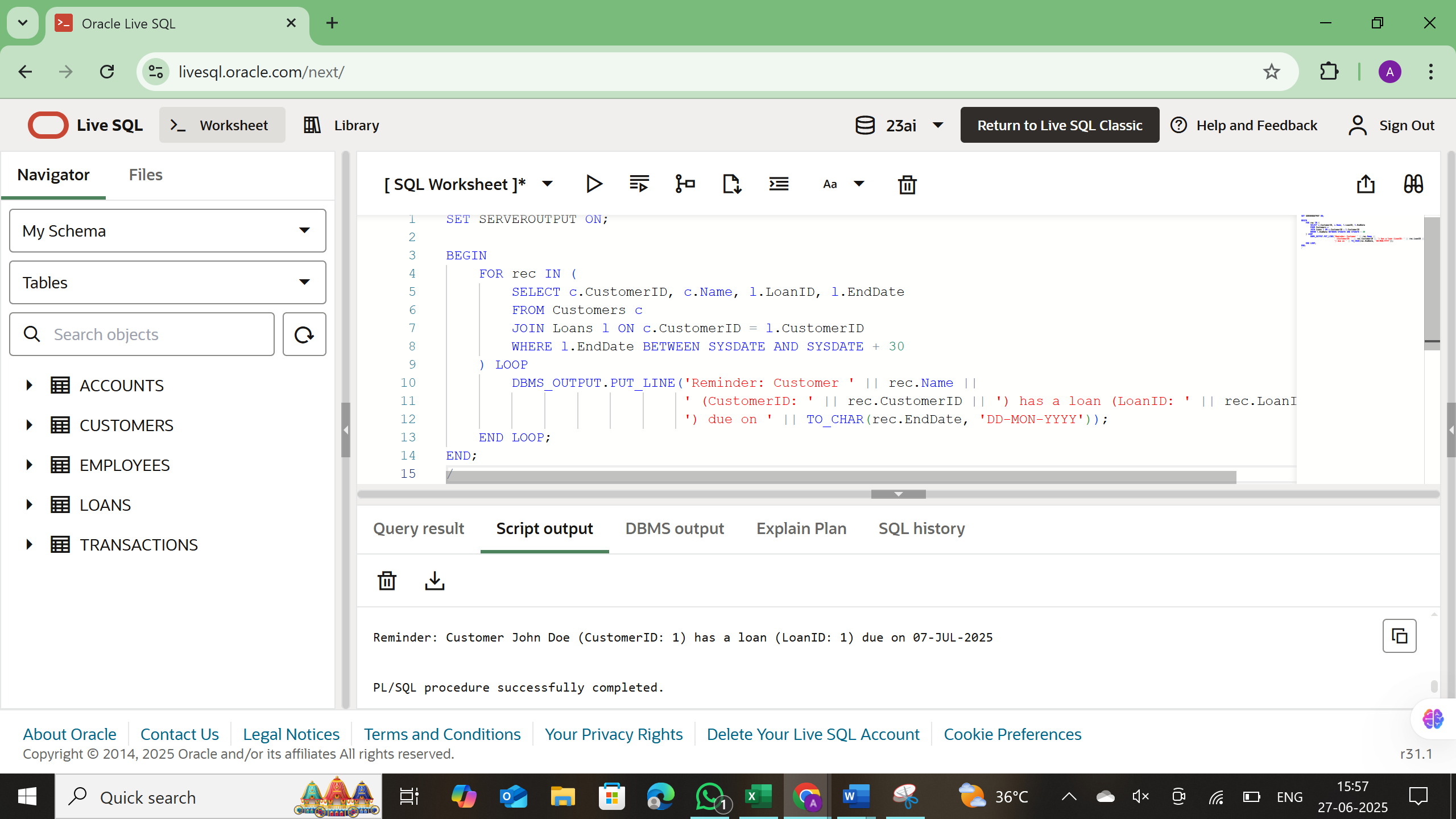
*DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ' || rec.Name ||*

*' (CustomerID: ' || rec.CustomerID || ') has a loan (LoanID: ' || rec.LoanID ||*

*') due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));*

*END LOOP;*

*END;*



***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:*** *CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS*

*BEGIN*

*UPDATE Accounts*

*SET Balance = Balance + (Balance \* 0.01)*

*WHERE AccountType = 'Savings';*

*COMMIT;*

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

*END;*

*/*

A screenshot of a computer

AI-generated content may be incorrect.

*Output:*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:***

*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;*

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

*END;*

*/*

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:*

*CREATE OR REPLACE PROCEDURE TransferFunds (*

*from\_account IN NUMBER,*

*to\_account IN NUMBER,*

*amount IN NUMBER*

*) AS*

*from\_balance NUMBER;*

*BEGIN*

*-- Get balance of source account*

*SELECT Balance INTO from\_balance*

*FROM Accounts*

*WHERE AccountID = from\_account;*

*-- Check if balance is sufficient*

*IF from\_balance < amount THEN*

*DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in AccountID: ' || from\_account);*

*RETURN;*

*END IF;*

*-- Deduct from source*

*UPDATE Accounts*

*SET Balance = Balance - amount*

*WHERE AccountID = from\_account;*

*-- Add to target*

*UPDATE Accounts*

*SET Balance = Balance + amount*

*WHERE AccountID = to\_account;*

*COMMIT;*

*DBMS\_OUTPUT.PUT\_LINE('Transferred ₹' || amount || ' from AccountID ' || from\_account || ' to AccountID ' || to\_account);*

*END;*

*/*

A screenshot of a computer

AI-generated content may be incorrect.