**Exercise 3: Stored Procedures**

**Initial Schema Given**

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Balance NUMBER,

    LastModified DATE

);

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

    TransactionID NUMBER PRIMARY KEY,

    AccountID NUMBER,

    TransactionDate DATE,

    Amount NUMBER,

    TransactionType VARCHAR2(10),

    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    LoanAmount NUMBER,

    InterestRate NUMBER,

    StartDate DATE,

    EndDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    Position VARCHAR2(50),

    Salary NUMBER,

    Department VARCHAR2(50),

    HireDate DATE

);

**Values inserted Initially**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (3, 'Nakka Girish', TO\_DATE('1960-12-20', 'YYYY-MM-DD'), 25000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 15000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3, 3, 'Savings', 25000, SYSDATE);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (3, 3, SYSDATE, 500, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (2, 3, 6000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 90));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (3, 'Nakka Girish', 'Java\_FSE', 90000, 'IT', TO\_DATE('2024-08-20', 'YYYY-MM-DD'));

Table values Initially

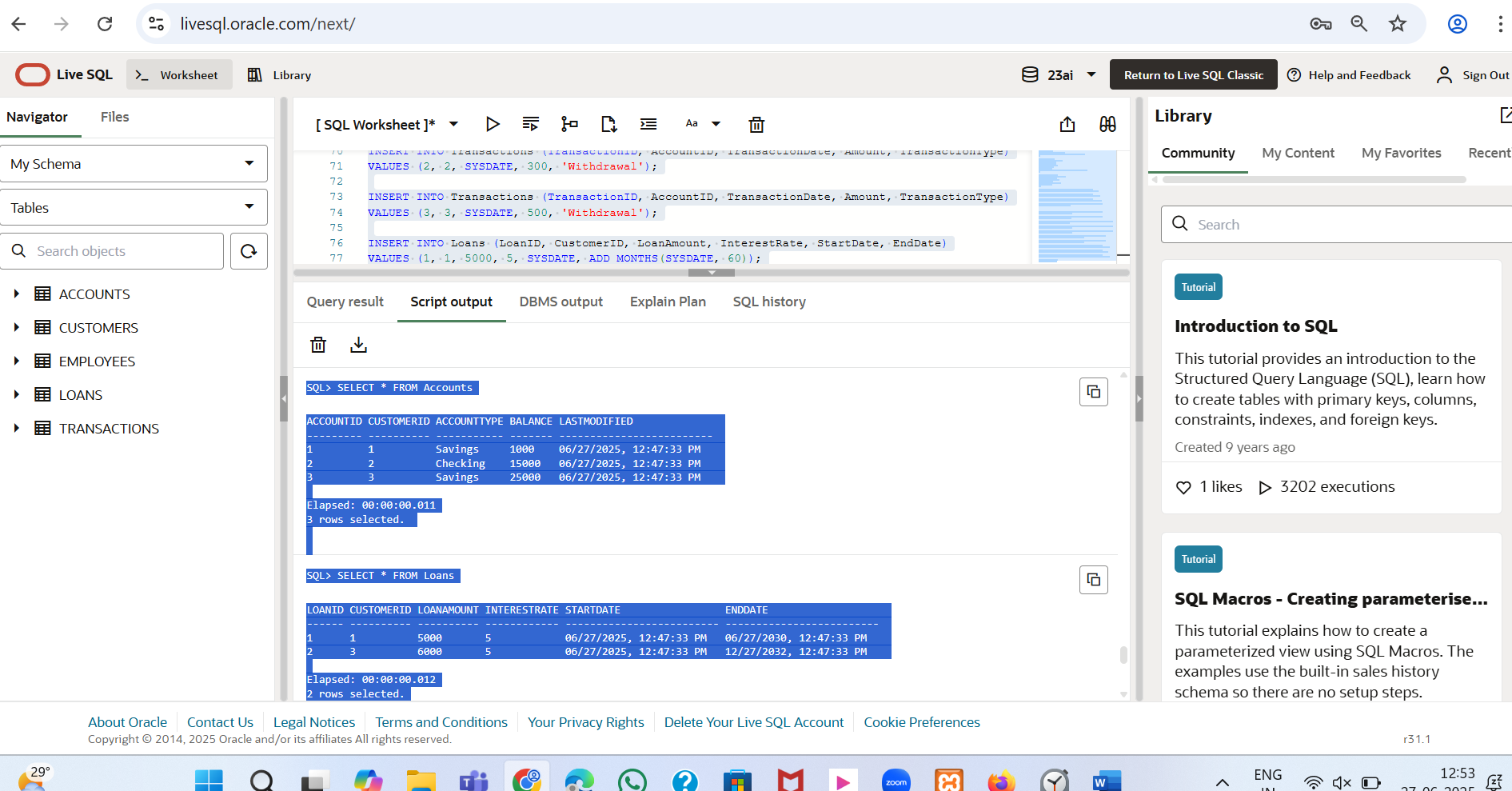
SELECT \* FROM Customers;

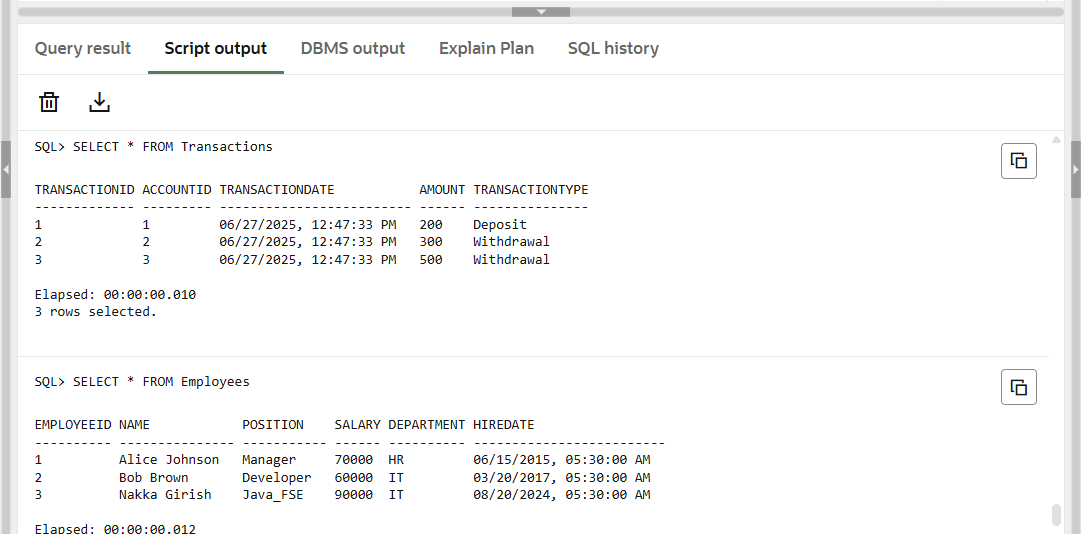
SELECT \* FROM Accounts;

SELECT \* FROM Loans;

SELECT \* FROM Transactions;

SELECT \* FROM Employees;

****

****

**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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

    UPDATE Accounts

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

    WHERE AccountID = acc.AccountID;

    DBMS\_OUTPUT.PUT\_LINE(

      'Interest applied to Account ID: ' || acc.AccountID ||

      ', new balance is: ' || TO\_CHAR(acc.Balance + (acc.Balance \* 0.01)));

  END LOOP;

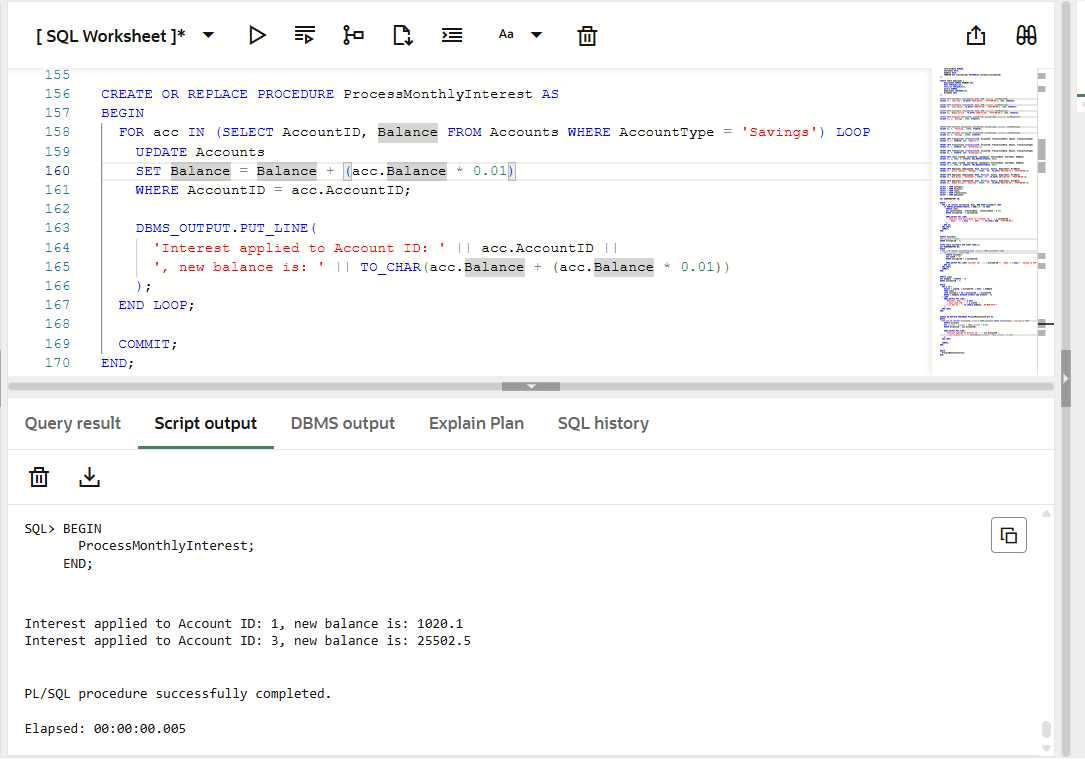
  COMMIT;

END;

BEGIN

  ProcessMonthlyInterest;

END;

**Output:** ****

**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.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  dept IN VARCHAR2,

  bonus\_percent IN NUMBER

) AS

BEGIN

  FOR emp IN (SELECT EmployeeID, Salary, Name FROM Employees WHERE Department = dept) LOOP

    UPDATE Employees

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

    WHERE EmployeeID = emp.EmployeeID;

    DBMS\_OUTPUT.PUT\_LINE(

      'Bonus applied to Employee ID: ' || emp.EmployeeID ||

      ' | Name: ' || emp.Name ||

      ' | New Salary: ' || TO\_CHAR(emp.Salary + (emp.Salary \* (bonus\_percent / 100)));

  END LOOP;

  COMMIT;

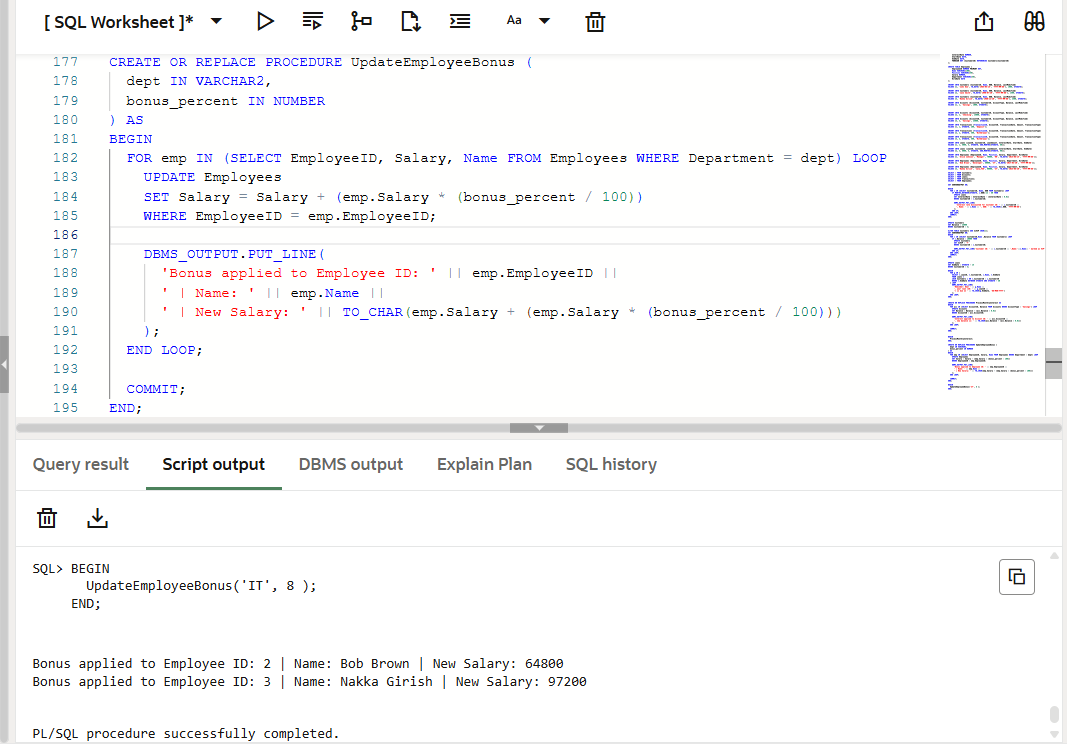
END;

BEGIN

  UpdateEmployeeBonus('IT', 8 );

END;

Output:



**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.

CREATE OR REPLACE PROCEDURE TransferFunds (

  from\_acc IN NUMBER,

  to\_acc IN NUMBER,

  amount IN NUMBER

) AS

  from\_balance NUMBER;

BEGIN

  SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = from\_acc;

  IF from\_balance >= amount THEN

    UPDATE Accounts

    SET Balance = Balance - amount

    WHERE AccountID = from\_acc;

    UPDATE Accounts

    SET Balance = Balance + amount

    WHERE AccountID = to\_acc;

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

    COMMIT;

  ELSE

    DBMS\_OUTPUT.PUT\_LINE('Insufficient amt in Account ID ' || from\_acc);

  END IF;

END;

BEGIN

  TransferFunds(1, 3, 500);

END;

**Output:**

