PL/SQL EXERCISE:

DATABASE:

**1.CUSTOMER TABLE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CUSTOMERID | NAME | DOB | BALANCE | LASTMODIFIED | ISVIP |
| 1 | "John Doe" | "1985-05-15T00:00:00Z" | 1000 | "2025-06-29T13:14:02Z" | "FALSE" |
| 3 | "Senior Sam" | "1940-01-01T00:00:00Z" | 2500 | "2025-06-29T13:53:02Z" | "FALSE" |
| 2 | "Jane Smith" | "1990-07-20T00:00:00Z" | 1500 | "2025-06-29T13:12:40Z" | "FALSE" |
| 4 | "Vinay" | "1960-01-01T00:00:00Z" | 15000 | "2025-06-29T13:58:02Z" | "TRUE" |

**2.ACCOUNTS TABLE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCOUNTID | CUSTOMERID | ACCOUNTTYPE | BALANCE | LASTMODIFIED |
| 1 | 1 | "Savings" | 1000 | "2025-06-29T13:15:04Z" |
| 2 | 2 | "Checking" | 1500 | "2025-06-29T13:15:23Z" |

**3.EMPLOYEES TABLE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMPLOYEEID | NAME | POSITION | SALARY | DEPARTMENT | HIREDATE |
| 1 | "Alice Johnson" | "Manager" | 70000 | "HR" | "2015-06-15T00:00:00Z" |
| 2 | "Bob Brown" | "Developer" | 60000 | "IT" | "2017-03-20T00:00:00Z" |

**4. LOANS TABLE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LOANID | CUSTOMERID | LOANAMOUNT | INTERESTRATE | STARTDATE | ENDDATE |
| 1 | 1 | 5000 | 5 | "2025-06-29T13:17:06Z" | "2030-06-29T13:17:06Z" |
| 5 | 1 | 3000 | 7 | "2024-07-04T14:02:13Z" | "2025-07-09T14:02:13Z" |
| 2 | 3 | 8000 | 6 | "2025-06-29T13:53:02Z" | "2030-06-29T13:53:02Z" |

**5. TRANSACTIONS TABLE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TRANSACTIONID | ACCOUNTID | TRANSACTIONDATE | AMOUNT | TRANSACTIONTYPE |
| 1 | 1 | "2025-06-29T13:16:22Z" | 200 | "Deposit" |
| 2 | 2 | "2025-06-29T13:16:22Z" | 300 | "Withdrawal" |

**Exercise 1: Control Structures**

**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.

**SOLUTION:**

SET SERVEROUTPUT ON;

DECLARE

   CURSOR seniors IS

      SELECT c.CustomerID, l.InterestRate

      FROM Customers c

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

      WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60;

BEGIN

   FOR rec IN seniors LOOP

      UPDATE Loans

      SET InterestRate = InterestRate - 1

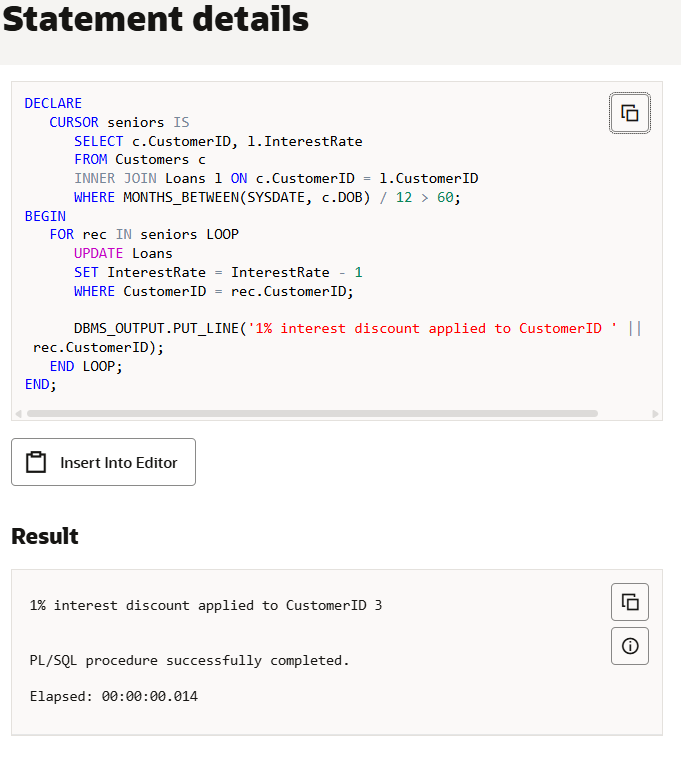
      WHERE CustomerID = rec.CustomerID;

      DBMS\_OUTPUT.PUT\_LINE('1% interest discount applied to CustomerID ' || rec.CustomerID);

   END LOOP;

END;

**OUTPUT:**

****

**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.

**SOLUTION:**

BEGIN

   FOR cust IN (

      SELECT CustomerID, Balance FROM Customers

   ) LOOP

      IF cust.Balance > 10000 THEN

         UPDATE Customers

         SET IsVIP = 'TRUE'

         WHERE CustomerID = cust.CustomerID;

         DBMS\_OUTPUT.PUT\_LINE('CustomerID ' || cust.CustomerID || ' promoted to VIP!');

      ELSE

         UPDATE Customers

         SET IsVIP = 'FALSE'

         WHERE CustomerID = cust.CustomerID;

      END IF;

   END LOOP;

END;

**OUTPUT:**

****

**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.

**SOLUTION:**

SET SERVEROUTPUT ON;

BEGIN

   FOR loan\_rec IN (

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

      FROM Loans l

      JOIN Customers c ON l.CustomerID = c.CustomerID

      WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

   ) LOOP

      DBMS\_OUTPUT.PUT\_LINE(

         'Reminder: LoanID ' || loan\_rec.LoanID ||

         ' for customer "' || loan\_rec.Name ||

         '" is due on ' || TO\_CHAR(loan\_rec.EndDate, 'DD-Mon-YYYY')

      );

   END LOOP;

END;

**OUTPUT:**

****

**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.

**SOLUTION:**

**SET SERVEROUTPUT ON;**

-- 1. Create the stored procedure

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

   FOR acc IN (

      SELECT AccountID, Balance

      FROM Accounts

      WHERE AccountType = 'Savings'

   ) LOOP

      UPDATE Accounts

      SET Balance = acc.Balance \* 1.01

      WHERE AccountID = acc.AccountID;

      DBMS\_OUTPUT.PUT\_LINE('Interest applied to AccountID: ' || acc.AccountID);

   END LOOP;

END;

/

-- 2. Execute the stored procedure

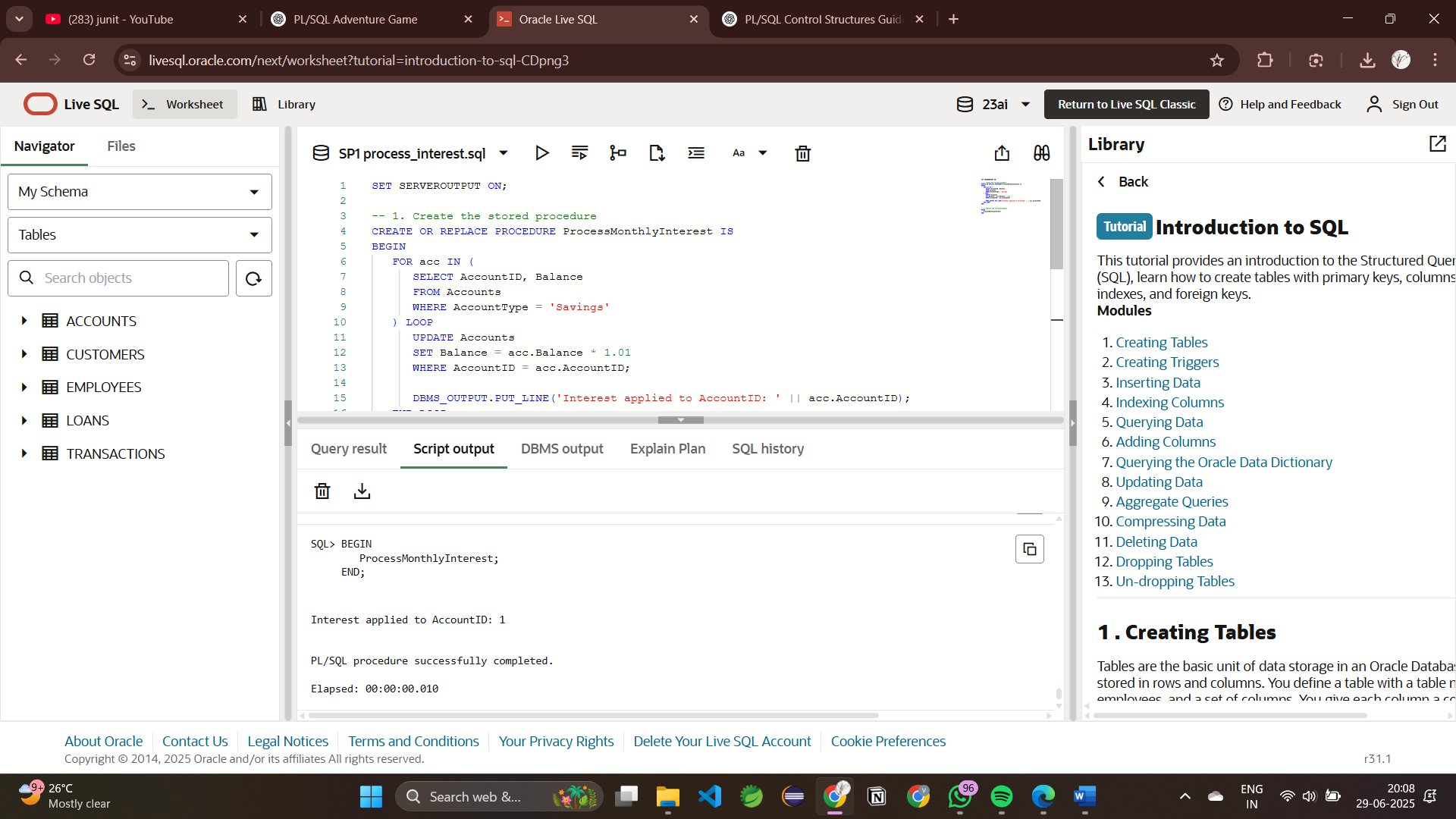
BEGIN

   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.

**SOLUTION:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

   dept IN VARCHAR2,

   bonus IN NUMBER

) IS

BEGIN

   FOR emp IN (

      SELECT EmployeeID, Name, Salary FROM Employees

      WHERE Department = dept

   ) LOOP

      UPDATE Employees

      SET Salary = Salary + (Salary \* bonus / 100)

      WHERE EmployeeID = emp.EmployeeID;

      DBMS\_OUTPUT.PUT\_LINE(

         ' Bonus applied to ' || emp.Name ||

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

  );

   END LOOP;

END;

/

SET SERVEROUTPUT ON;

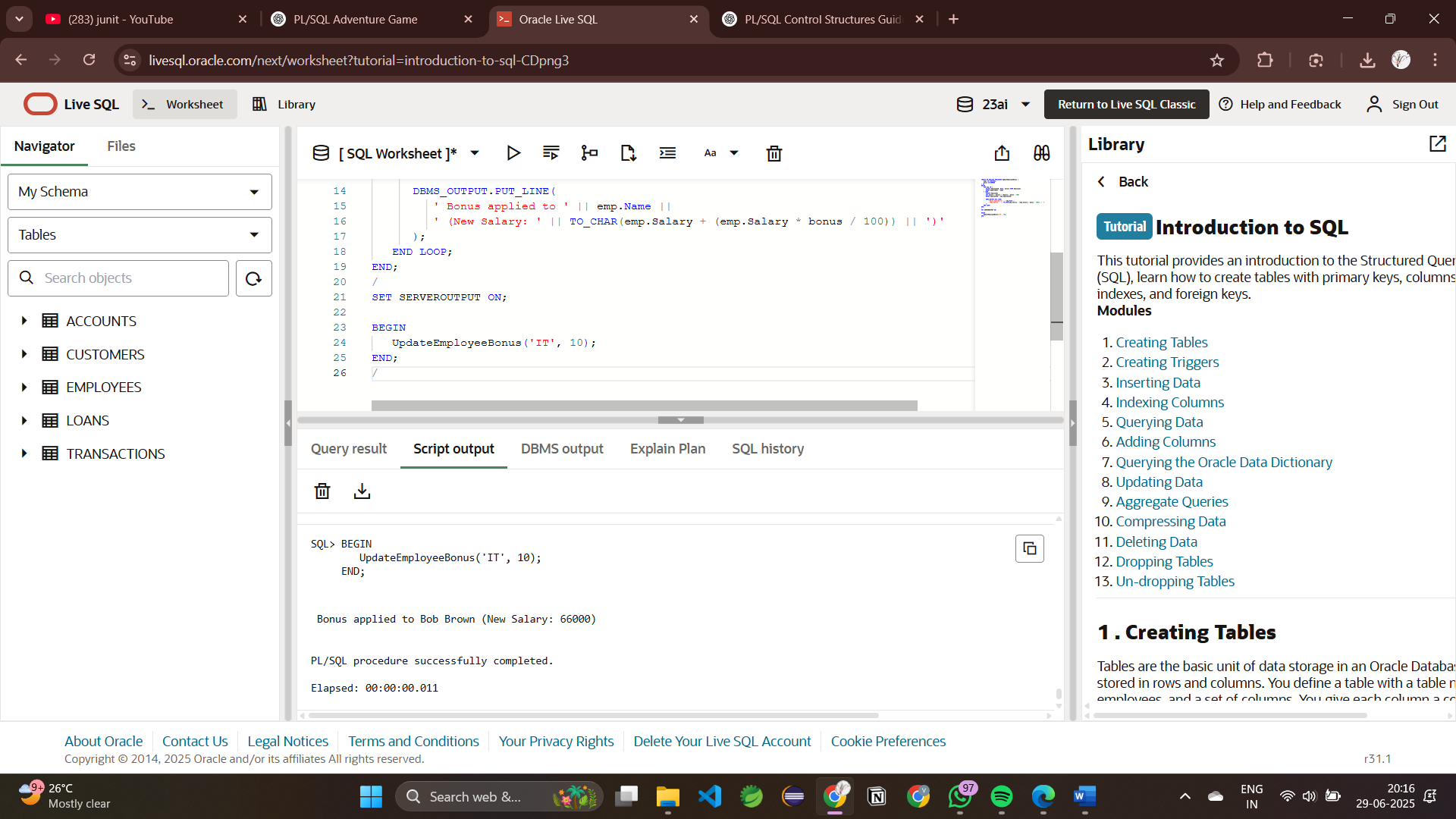
BEGIN

   UpdateEmployeeBonus('IT', 10);

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.

**SOLUTION:**

-- Procedure: TransferFunds

-- Purpose: Transfers funds between two accounts if sufficient balance is available

CREATE OR REPLACE PROCEDURE TransferFunds (

   p\_source\_id     IN NUMBER,   -- Source Account ID

   p\_dest\_id       IN NUMBER,   -- Destination Account ID

   p\_amount        IN NUMBER    -- Amount to transfer

) IS

   v\_source\_balance NUMBER;     -- Variable to store current balance of source account

BEGIN

   -- Step 1: Fetch balance of source account

   SELECT Balance INTO v\_source\_balance

   FROM Accounts

   WHERE AccountID = p\_source\_id;

   -- Step 2: Check if balance is sufficient

   IF v\_source\_balance < p\_amount THEN

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

   ELSE

      -- Step 3: Deduct the amount from the source account

      UPDATE Accounts

      SET Balance = Balance - p\_amount

      WHERE AccountID = p\_source\_id;

      -- Step 4: Add the amount to the destination account

      UPDATE Accounts

      SET Balance = Balance + p\_amount

      WHERE AccountID = p\_dest\_id;

      -- Step 5: Confirmation message

      DBMS\_OUTPUT.PUT\_LINE('Transfer of Rs.' || p\_amount ||

                           ' completed from Account ' || p\_source\_id ||

                           ' to Account ' || p\_dest\_id);

END IF;

EXCEPTION

   WHEN NO\_DATA\_FOUND THEN

      DBMS\_OUTPUT.PUT\_LINE('Error: One of the account IDs does not exist.');

   WHEN OTHERS THEN

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

END;

/

SET SERVEROUTPUT ON;

BEGIN

   TransferFunds(1, 2, 200);

END;

/

**OUTPUT:**

