Exercise 1: Control Structures

TABLE :

```
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
);
```

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:

```
BEGIN
   FOR rec IN (SELECT c.CustomerID, l.LoanID, l.InterestRate,
TRUNC(MONTHS_BETWEEN(SYSDATE, c.DOB) / 12) AS Age
            FROM Customers c JOIN Loans l ON c.CustomerID = l.CustomerID)
   LOOP
      IF rec.Age > 60 THEN
```

```
        UPDATE Loans
        SET InterestRate = InterestRate - 1
        WHERE LoanID = rec.LoanID;
      END IF;
  END LOOP;
END;
/
```

BEFORE SCENARIO 1:

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.085 seconds

	LOANID	CUSTOMERID	LOANAMOUNT	INTERESTRATE	STARTDATE	ENDDATE
1	1	1	5000	5	6/29/2025, 3:01:56	6/29/2030, 3:01:56
2	2	2	8000	6	6/29/2025, 3:01:56	6/29/2028, 3:01:56

AFTER SCENARIO 1 (OUTPUT):

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.007 seconds

	LOANID	CUSTOMERID	LOANAMOUNT	INTERESTRATE	STARTDATE	ENDDATE
1	1	1	5000	5	6/29/2025, 2:58:01	6/29/2030, 2:58:01
2	2	2	8000	5	6/29/2025, 2:58:01	6/29/2028, 2:58:01

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:

```
ALTER TABLE Customers ADD IsVIP VARCHAR2(5);
BEGIN
  FOR rec IN (SELECT CustomerID, Balance FROM Customers)
  LOOP
```

```
        IF rec.Balance > 10000 THEN
            UPDATE Customers SET IsVIP = 'TRUE' WHERE CustomerID = rec.CustomerID;
        END IF;
    END LOOP;
END;
/
```

BEFORE SCENARIO 2:

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.083 seconds

	CUSTOMERID	NAME	DOB	BALANCE	LASTMODIFIED
1	1	John Doe	5/15/1985, 12:00:0(	1000	6/29/2025, 3:01:10
2	2	Jane Smith	7/20/1960, 12:00:0(	15000	6/29/2025, 3:01:10

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.009 seconds

	CUSTOMERID	NAME	DOB	BALANCE	LASTMODIFIED	ISVIP
1	1	John Doe	5/15/1985, 12:00:0(	1000	6/29/2025, 3:01:10	*(null)*
2	2	Jane Smith	7/20/1960, 12:00:0(	15000	6/29/2025, 3:01:10	*(null)*

AFTER SCENARIO 2 (OUTPUT):

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.001 seconds

	CUSTOMERID	NAME	DOB	BALANCE	LASTMODIFIED	ISVIP
1	1	John Doe	5/15/1985, 12:00:0(	1000	6/29/2025, 2:52:36	*(null)*
2	2	Jane Smith	7/20/1960, 12:00:0(	15000	6/29/2025, 2:52:36	TRUE

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:

```
BEGIN
   FOR rec IN (
      SELECT l.LoanID, 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: ' || rec.Name || ' - Your loan is due on ' ||
TO_CHAR(rec.EndDate, 'DD-MON-YYYY'));
   END LOOP;
END;
/
```

AFTER SCENARIO 3 (OUTPUT):

Query result | **Script output** | DBMS output | Explain Plan | SQL history

```
SQL> BEGIN
        FOR rec IN (
            SELECT l.LoanID, c.Name, l.EndDate
            FROM Loans l...
Show more...

Reminder: Jane Smith, your loan is due on 2025-07-09

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008
```

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:

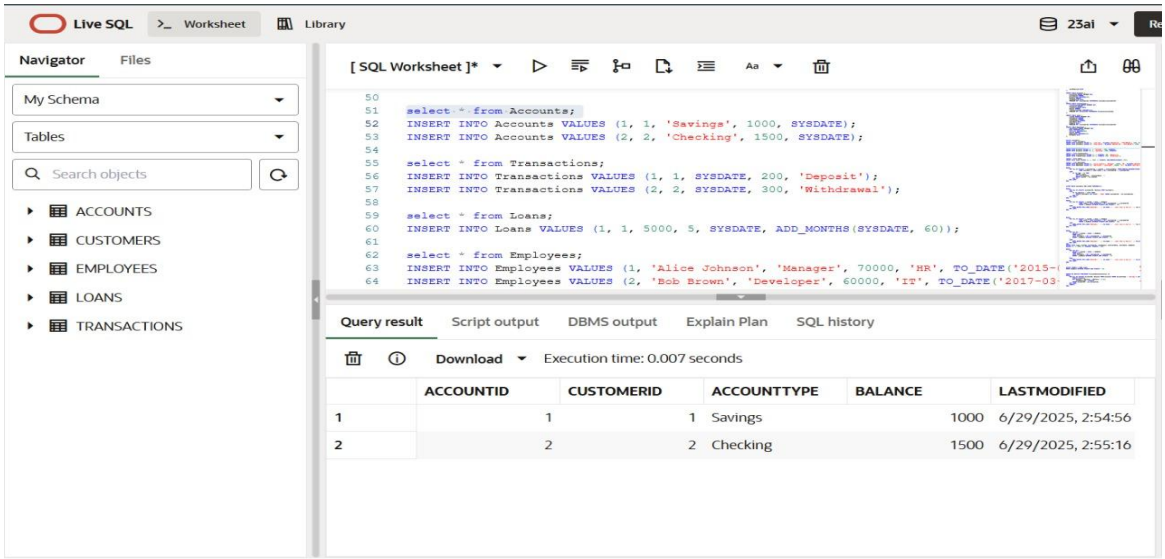
Procedure: ProcessMonthlyInterest

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
   FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP
      UPDATE Accounts
      SET Balance = Balance + (Balance * 0.01),
         LastModified = SYSDATE
      WHERE AccountID = acc.AccountID;
   END LOOP;
   COMMIT;
END; /
```

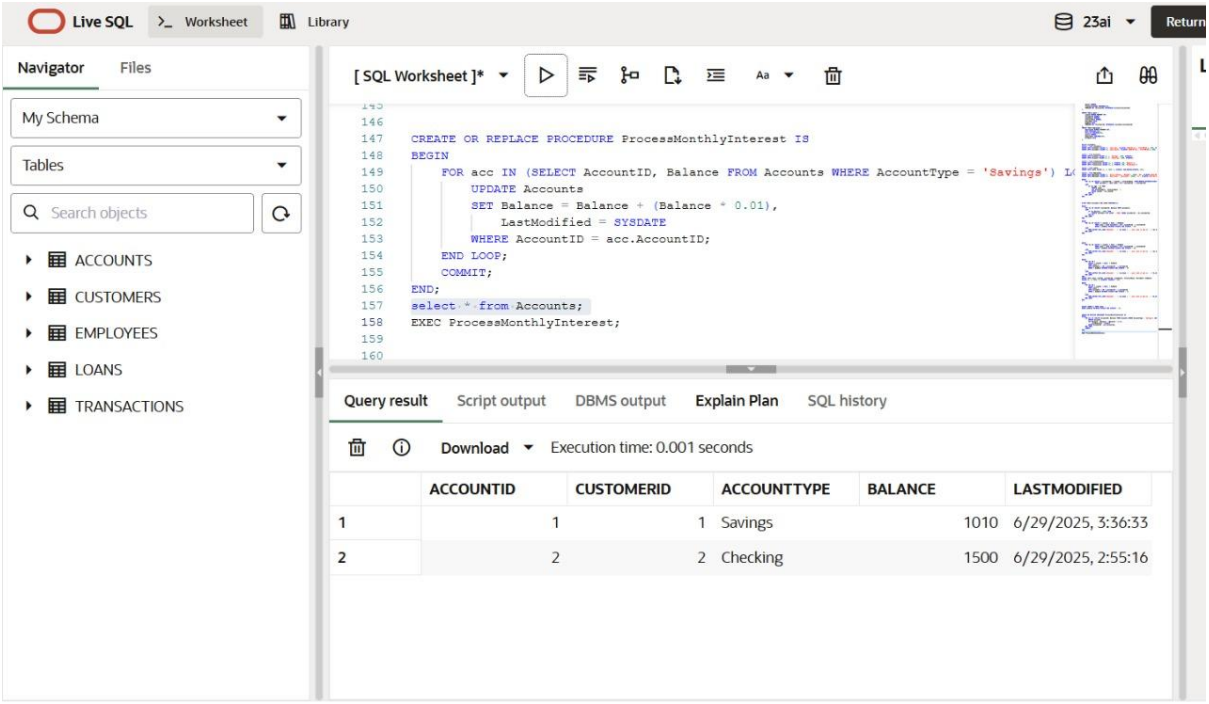
Usage:

```
EXEC ProcessMonthlyInterest;
```

BEFORE SCENARIO 1:



AFTER SCENARIO 1 (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.

CODE:

Procedure: UpdateEmployeeBonus

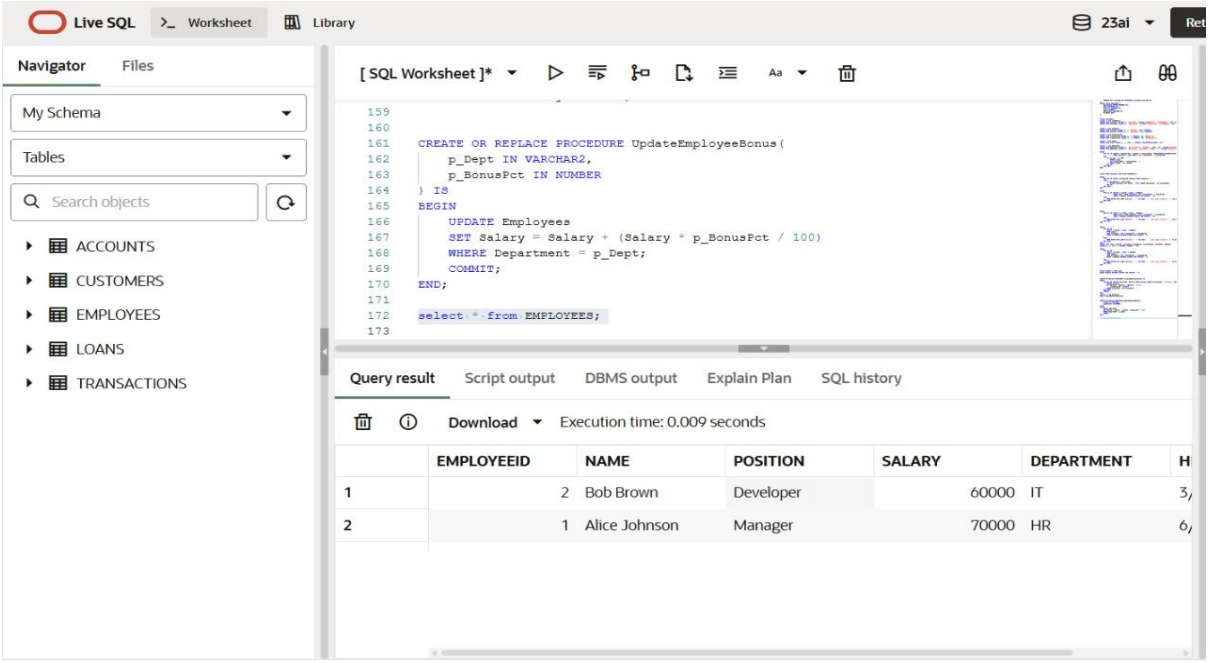
```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
   p_Dept IN VARCHAR2,
   p_BonusPct IN NUMBER
) IS
BEGIN
   UPDATE Employees
   SET Salary = Salary + (Salary * p_BonusPct / 100)
   WHERE Department = p_Dept;
   COMMIT;
END; /
```

Usage Example:

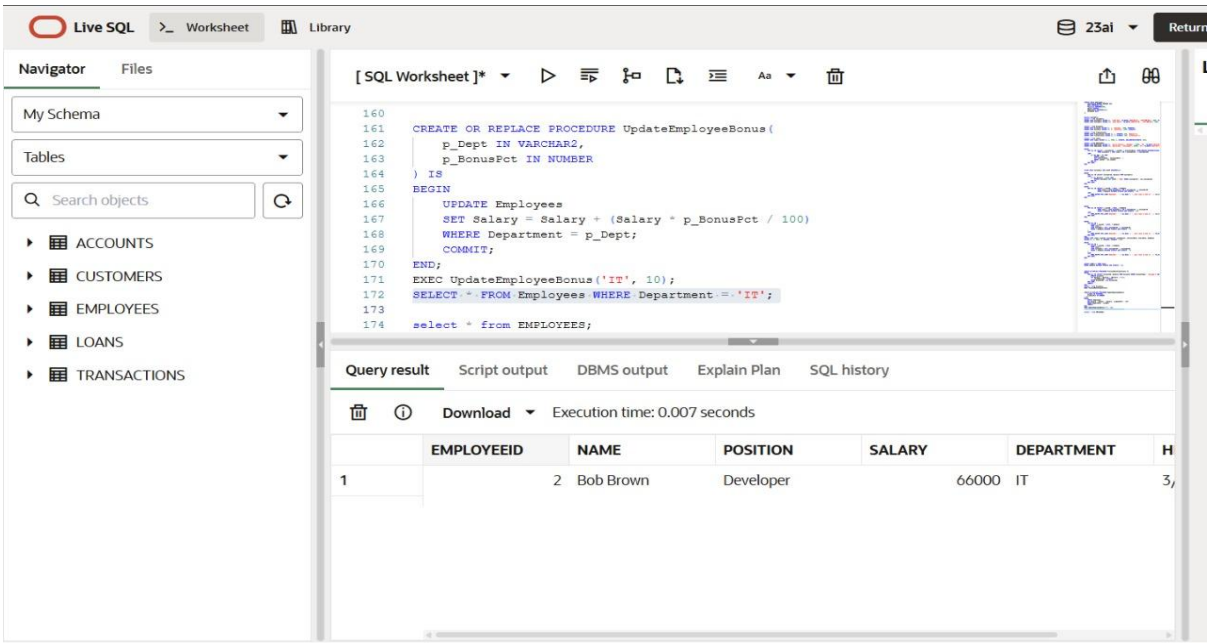
EXEC UpdateEmployeeBonus('IT', 10);

SELECT * FROM Employees WHERE Department = 'IT';

BEFORE SCENARIO 2:



AFTER SCENARIO 2 (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.

CODE:

Procedure: TransferFunds

```
CREATE OR REPLACE PROCEDURE TransferFunds(
   p_FromAccID IN NUMBER,
   p_ToAccID IN NUMBER,
   p_Amount IN NUMBER
) IS
   v_Balance NUMBER;
BEGIN
   SELECT Balance INTO v_Balance
   FROM Accounts
   WHERE AccountID = p_FromAccID
   FOR UPDATE;

   IF v_Balance >= p_Amount THEN
      -- Deduct from source
      UPDATE Accounts
      SET Balance = Balance - p_Amount,
         LastModified = SYSDATE
      WHERE AccountID = p_FromAccID;

      -- Add to destination
      UPDATE Accounts
      SET Balance = Balance + p_Amount,
         LastModified = SYSDATE
```

```
        WHERE AccountID = p_ToAccID;

        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Transfer successful.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Transfer failed: Insufficient balance.');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Transfer failed: Invalid Account ID.');
        ROLLBACK;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);
        ROLLBACK;
END;
/
```

Usage:

```
SET SERVEROUTPUT ON;
EXEC TransferFunds(1, 2, 500);  -- Transfer ₹500 from AccountID 1 to 2

SELECT * FROM Accounts WHERE AccountID IN (1, 2);
```

AFTER SCENARIO 3 (OUTPUT):