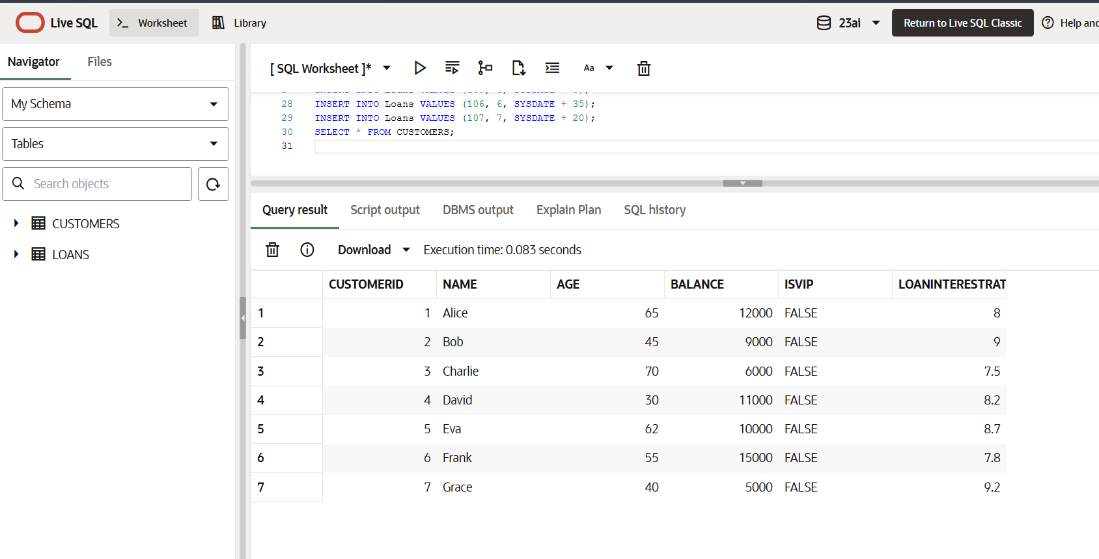
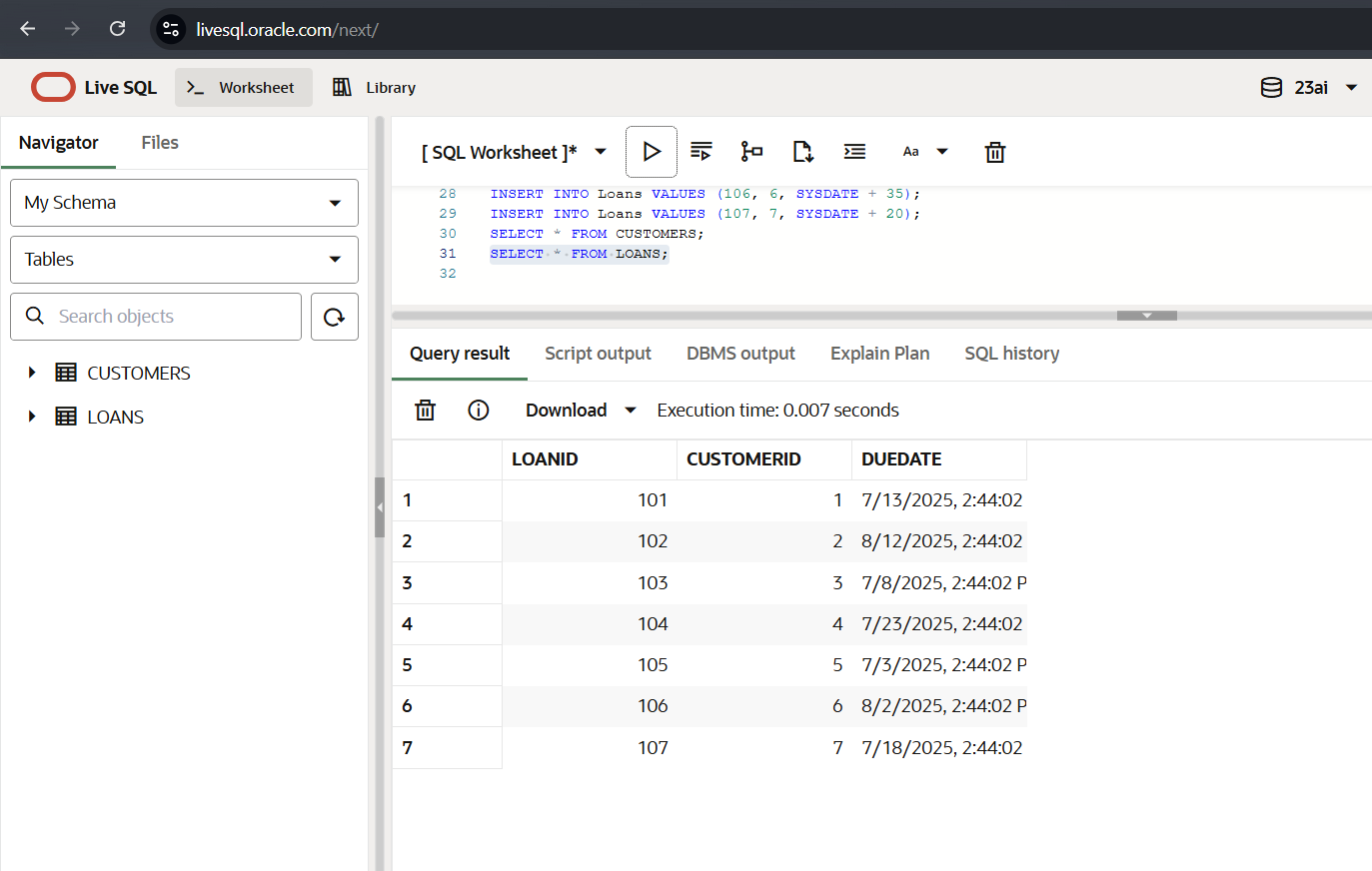
**Module 3 - PL/SQL Programming**

**Exercise 1: Control Structures**

|  |  |
| --- | --- |
| #Create Customers table and Insert Data  CREATE TABLE Customers (    CustomerID NUMBER PRIMARY KEY,    Name VARCHAR2(100),    Age NUMBER,    Balance NUMBER,    IsVIP VARCHAR2(5),    LoanInterestRate NUMBER  );  --Insert values  INSERT INTO Customers VALUES (1, 'Alice', 65, 12000, 'FALSE', 8.0);  INSERT INTO Customers VALUES (2, 'Bob', 45, 9000, 'FALSE', 9.0);  INSERT INTO Customers VALUES (3, 'Charlie', 70, 6000, 'FALSE', 7.5);  INSERT INTO Customers VALUES (4, 'David', 30, 11000, 'FALSE', 8.2);  INSERT INTO Customers VALUES (5, 'Eva', 62, 10000, 'FALSE', 8.7);  INSERT INTO Customers VALUES (6, 'Frank', 55, 15000, 'FALSE', 7.8);  INSERT INTO Customers VALUES (7, 'Grace', 40, 5000, 'FALSE', 9.2); | #Create Loans table and Insert Data  CREATE TABLE Loans (    LoanID NUMBER PRIMARY KEY,    CustomerID NUMBER REFERENCES Customers(CustomerID),    DueDate DATE  );  --Insert values  INSERT INTO Loans VALUES (101, 1, SYSDATE + 15);  INSERT INTO Loans VALUES (102, 2, SYSDATE + 45);  INSERT INTO Loans VALUES (103, 3, SYSDATE + 10);  INSERT INTO Loans VALUES (104, 4, SYSDATE + 25);  INSERT INTO Loans VALUES (105, 5, SYSDATE + 5);  INSERT INTO Loans VALUES (106, 6, SYSDATE + 35);  INSERT INTO Loans VALUES (107, 7, SYSDATE + 20); |

**OUTPUT:**

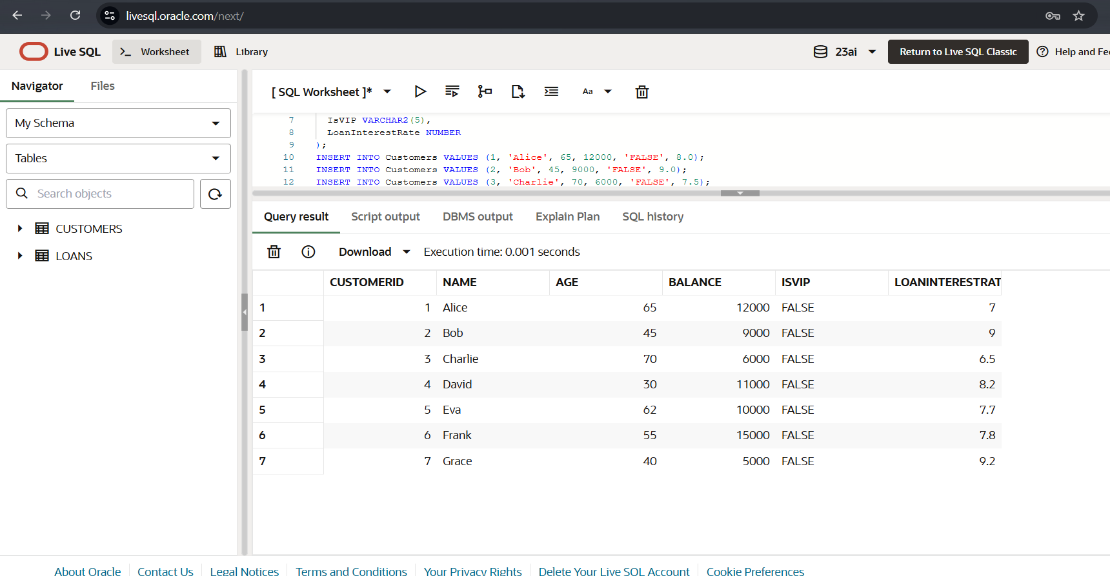
****

****

**Scenario 1: Query**

|  |
| --- |
| BEGIN    FOR cust IN (SELECT CustomerID, Age, LoanInterestRate FROM Customers)    LOOP      IF cust.Age > 60 THEN        UPDATE Customers        SET LoanInterestRate = LoanInterestRate - 1        WHERE CustomerID = cust.CustomerID;      END IF;    END LOOP;    COMMIT;  END; |

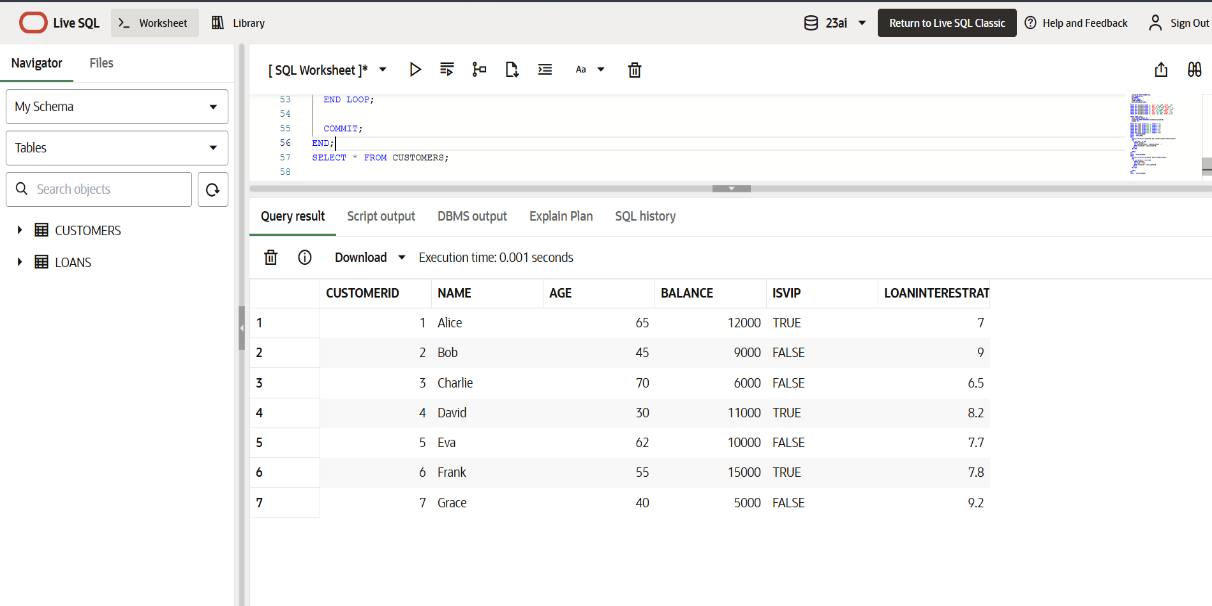
**OUTPUT:**

****

**Scenario 2: Query**

|  |
| --- |
| BEGIN    FOR cust IN (SELECT CustomerID, Balance FROM Customers)    LOOP      IF cust.Balance > 10000 THEN        UPDATE Customers        SET IsVIP = 'TRUE'        WHERE CustomerID = cust.CustomerID;      END IF;    END LOOP;    COMMIT;  END; |

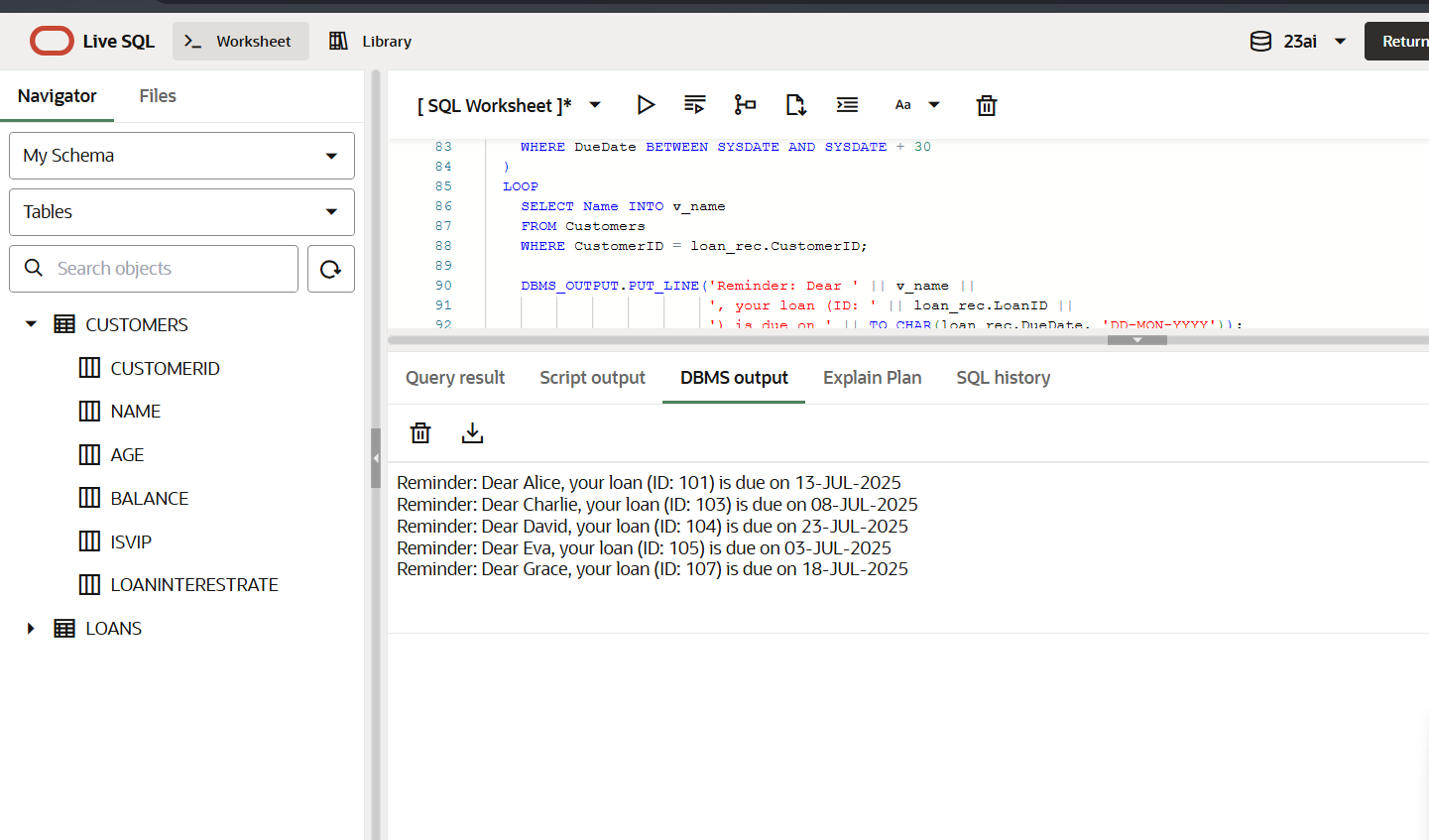
**OUTPUT:**

****

**Scenario 3: Query**

|  |
| --- |
| DECLARE  v\_name Customers.Name%TYPE;  BEGIN  FOR loan\_rec IN (  SELECT LoanID, CustomerID, DueDate  FROM Loans  WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30  )  LOOP  SELECT Name INTO v\_name  FROM Customers  WHERE CustomerID = loan\_rec.CustomerID;  DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_name ||  ', your loan (ID: ' || loan\_rec.LoanID ||  ') is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY'));  END LOOP;  END; |

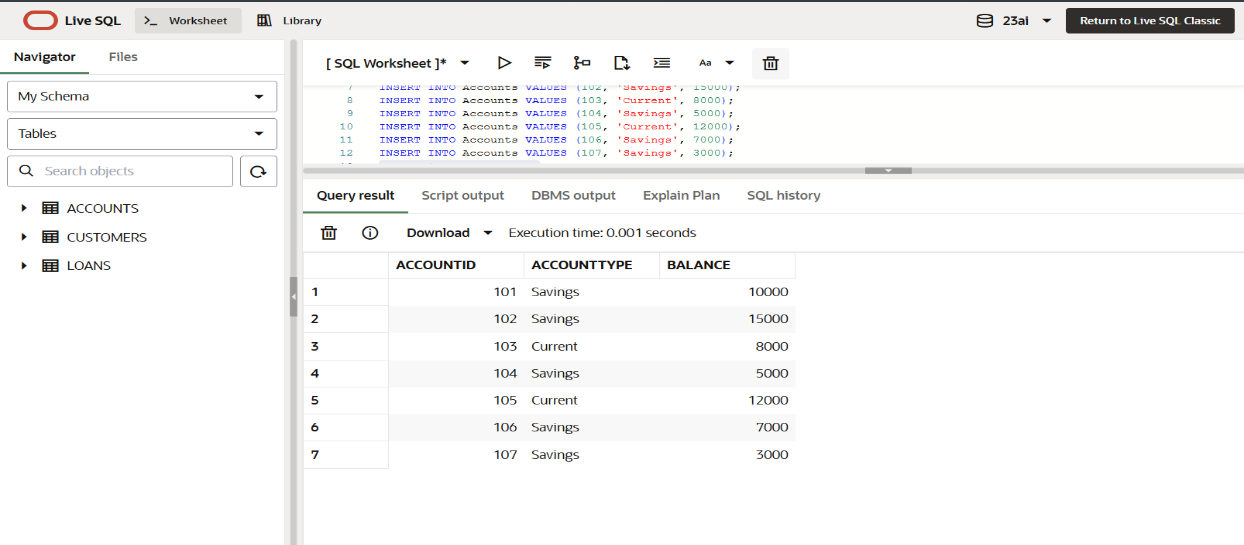
**OUTPUT:**

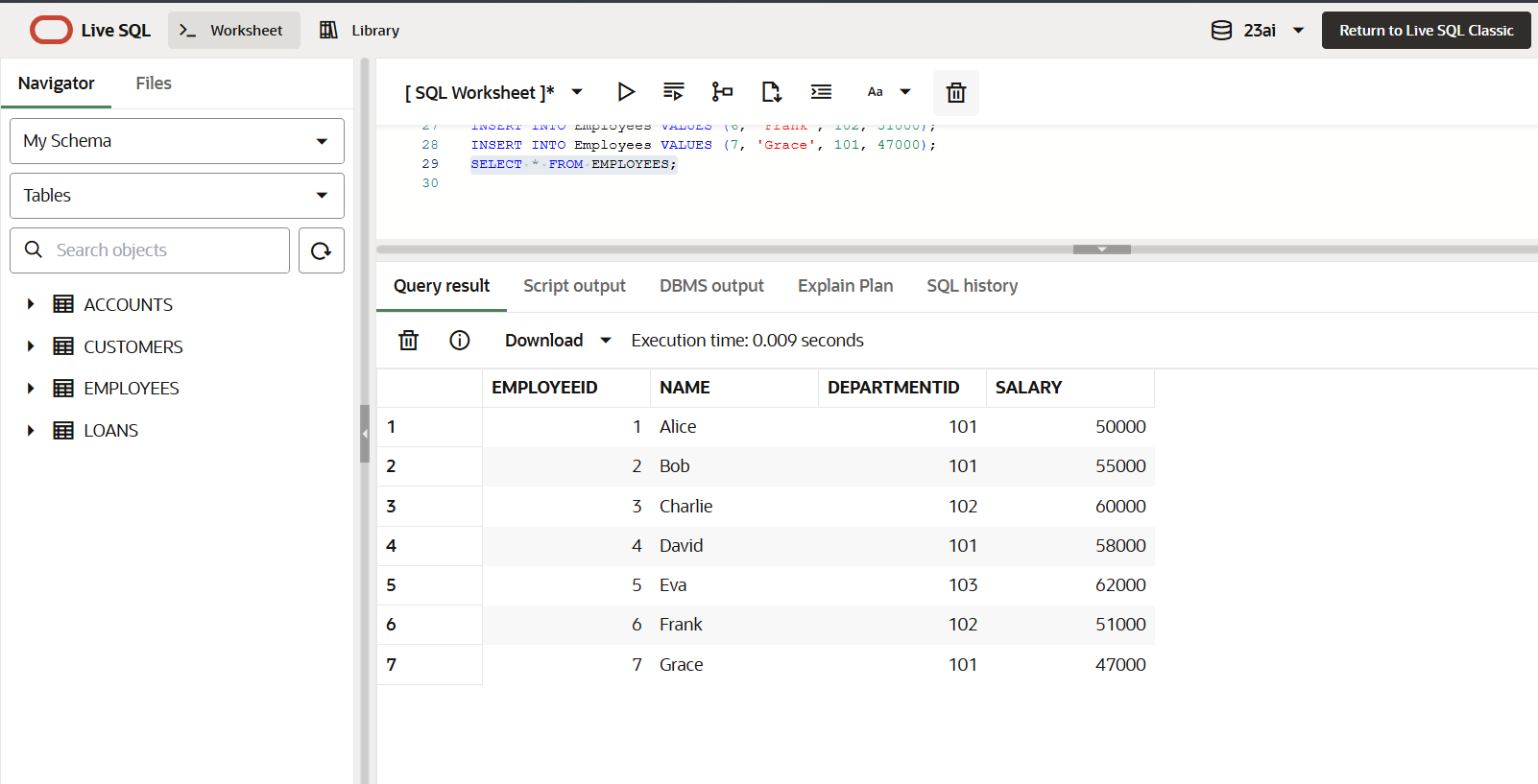
****

**Exercise 3: Stored Procedures**

|  |  |
| --- | --- |
| #Create Accounts Table  CREATE TABLE Accounts (    AccountID NUMBER PRIMARY KEY,    AccountType VARCHAR2(20),    Balance NUMBER  );  #Insert Values  INSERT INTO Accounts VALUES (101, 'Savings', 10000);  INSERT INTO Accounts VALUES (102, 'Savings', 15000);  INSERT INTO Accounts VALUES (103, 'Current', 8000);  INSERT INTO Accounts VALUES (104, 'Savings', 5000);  INSERT INTO Accounts VALUES (105, 'Current', 12000);  INSERT INTO Accounts VALUES (106, 'Savings', 7000);  INSERT INTO Accounts VALUES (107, 'Savings', 3000);  SELECT \* FROM ACCOUNTS; | #Create Employees Table  CREATE TABLE Employees (    EmployeeID NUMBER PRIMARY KEY,    Name VARCHAR2(100),    DepartmentID NUMBER,    Salary NUMBER  );  #Insert Values  INSERT INTO Employees VALUES (1, 'Alice', 101, 50000);  INSERT INTO Employees VALUES (2, 'Bob', 101, 55000);  INSERT INTO Employees VALUES (3, 'Charlie', 102, 60000);  INSERT INTO Employees VALUES (4, 'David', 101, 58000);  INSERT INTO Employees VALUES (5, 'Eva', 103, 62000);  INSERT INTO Employees VALUES (6, 'Frank', 102, 51000);  INSERT INTO Employees VALUES (7, 'Grace', 101, 47000);  SELECT \* FROM EMPLOYEES; |

**OUTPUT:**

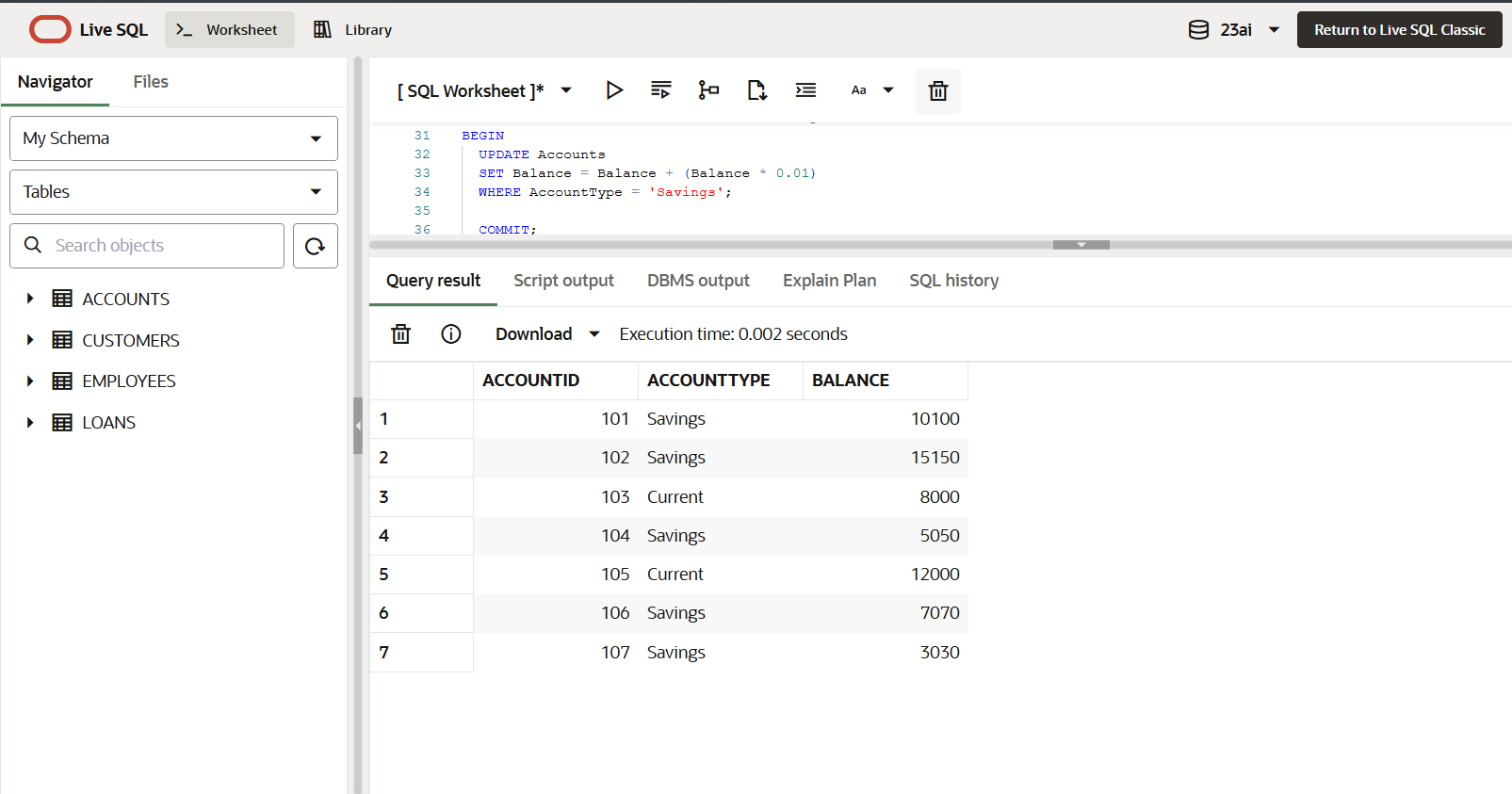
****

****

**Scenario 1: Query**

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS  BEGIN  UPDATE Accounts  SET Balance = Balance + (Balance \* 0.01)  WHERE AccountType = 'Savings';  COMMIT;  END;  BEGIN  ProcessMonthlyInterest;  END; |

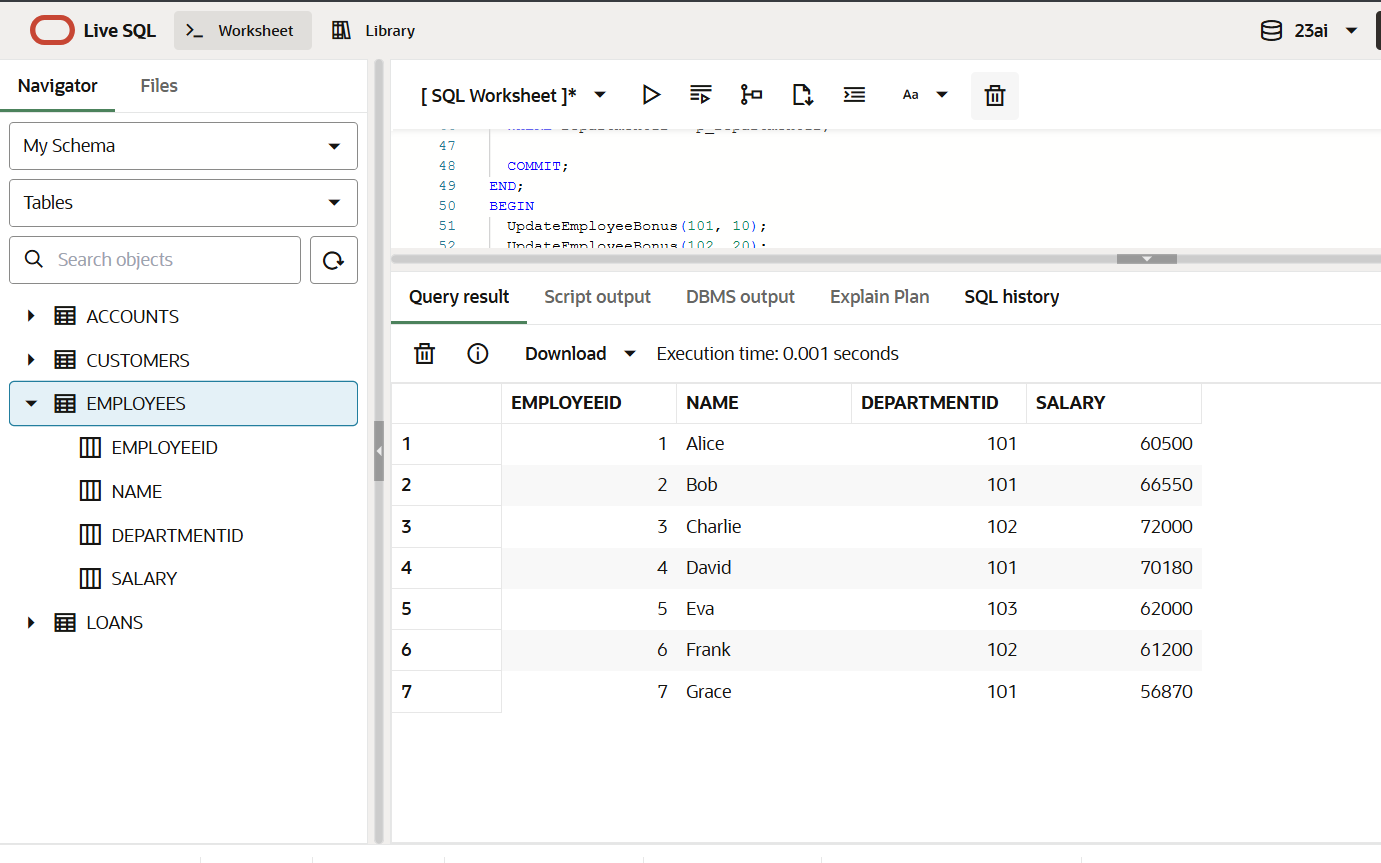
**OUTPUT:**

****

**Scenario 2: Query**

|  |
| --- |
| CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(    p\_DepartmentID IN NUMBER,    p\_BonusPercent IN NUMBER  ) IS  BEGIN    UPDATE Employees    SET Salary = Salary + (Salary \* p\_BonusPercent / 100)    WHERE DepartmentID = p\_DepartmentID;    COMMIT;  END;  BEGIN    UpdateEmployeeBonus(101, 10);  UpdateEmployeeBonus(102, 20);  END;  SELECT \* FROM EMPLOYEES; |

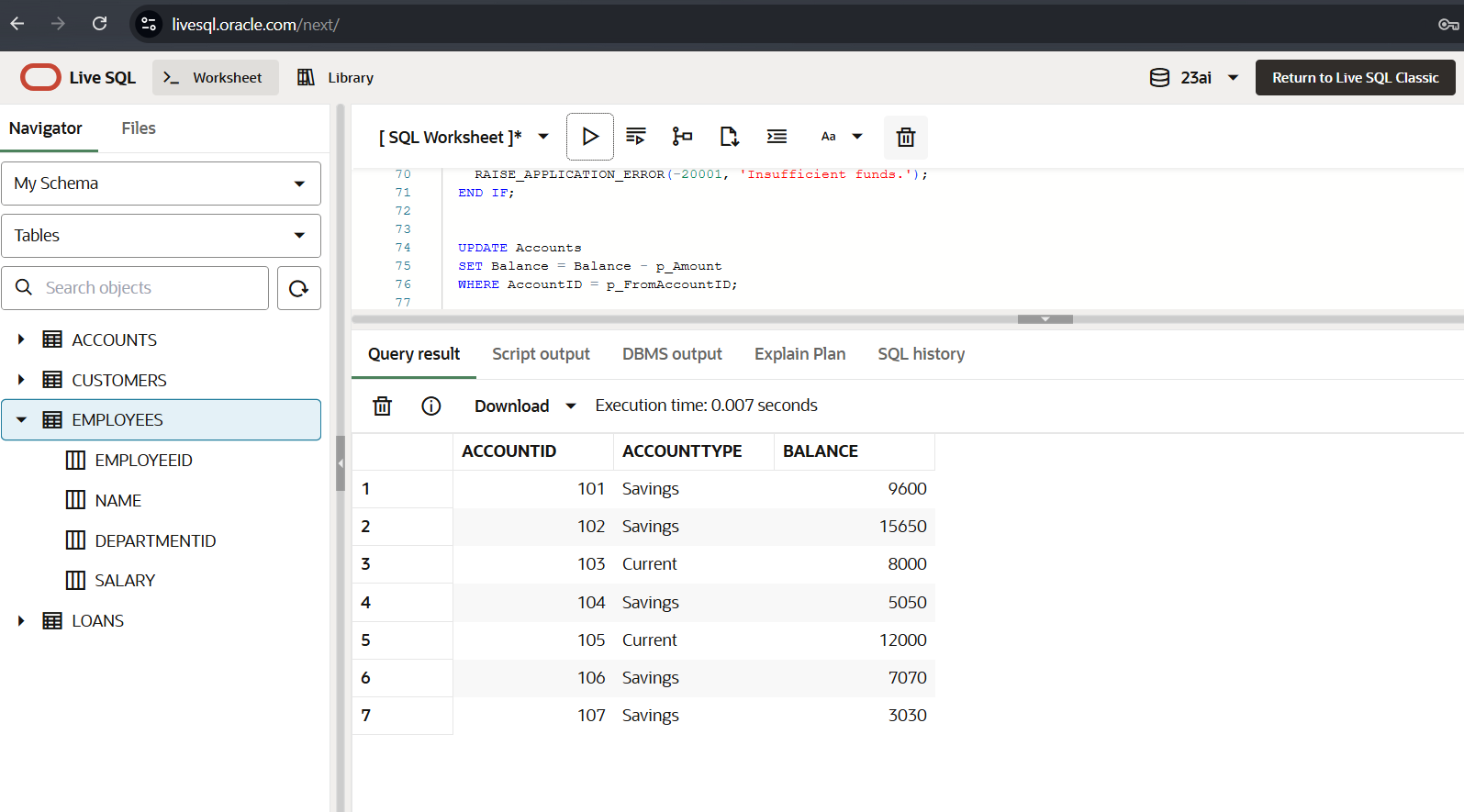
**OUTPUT:**

****

**Scenario 3: Query**

|  |
| --- |
| CREATE OR REPLACE PROCEDURE TransferFunds(    p\_FromAccountID IN NUMBER,    p\_ToAccountID IN NUMBER,    p\_Amount IN NUMBER  ) IS    v\_balance NUMBER;  BEGIN      SELECT Balance INTO v\_balance    FROM Accounts    WHERE AccountID = p\_FromAccountID;      IF v\_balance < p\_Amount THEN      RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds.');    END IF;      UPDATE Accounts    SET Balance = Balance - p\_Amount    WHERE AccountID = p\_FromAccountID;    UPDATE Accounts    SET Balance = Balance + p\_Amount    WHERE AccountID = p\_ToAccountID;    COMMIT;  END;  BEGIN    TransferFunds(101, 102, 500);  END;  SELECT \* FROM ACCOUNTS; |

**OUTPUT:**

****