**Week-2: (Module 3)**

**PL/SQL Programming**

**Exercise 3: Stored Procedures**

CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified 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

);

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

VALUES (1, 1, 'Savings', 10000, SYSDATE);

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

VALUES (2, 1, 'Checking', 8000, SYSDATE);

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

VALUES (3, 2, 'Savings', 5000, SYSDATE);

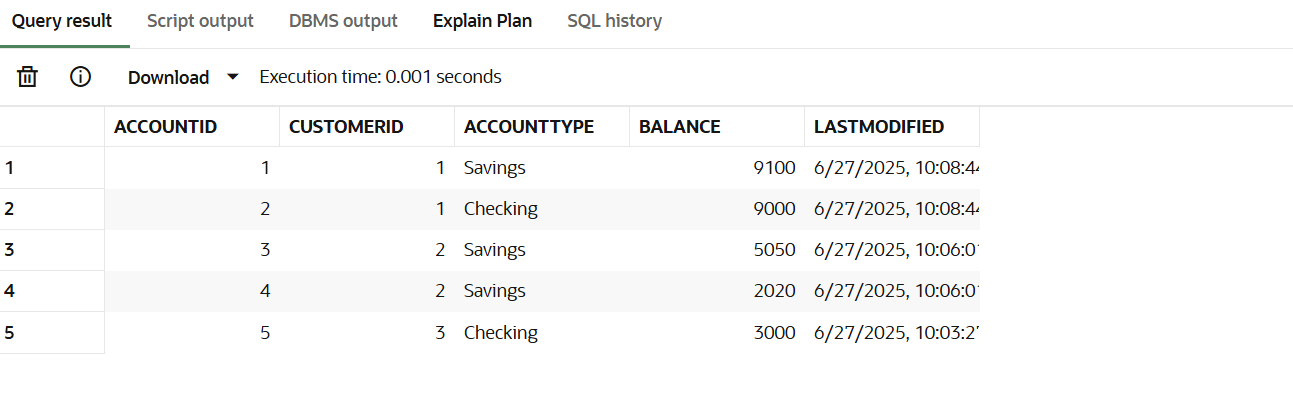
INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (4, 2, 'Savings', 2000, SYSDATE);

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

VALUES (5, 3, 'Checking', 3000, SYSDATE);

SELECT \* FROM Accounts;



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, 'Cathy Green', 'Analyst', 55000, 'Finance', TO\_DATE('2018-11-01', 'YYYY-MM-DD'));

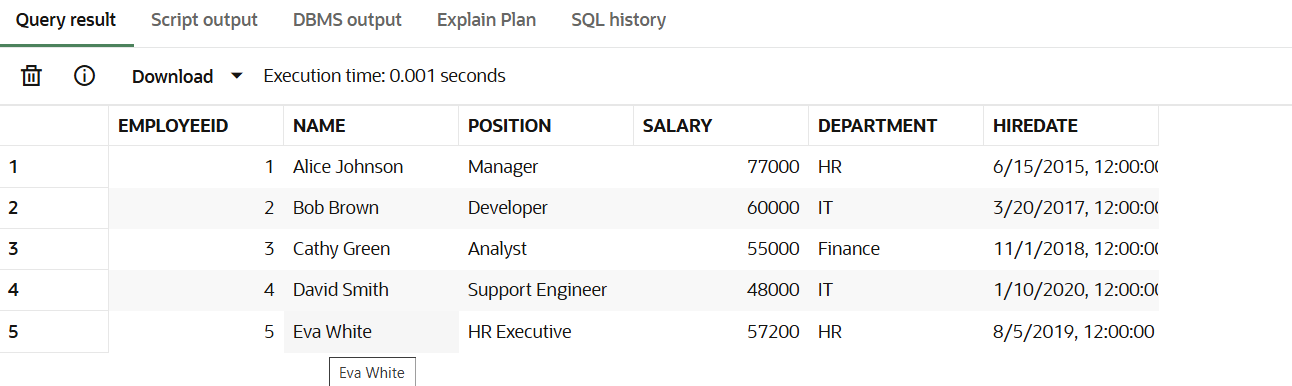
INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (4, 'David Smith', 'Support Engineer', 48000, 'IT', TO\_DATE('2020-01-10', 'YYYY-MM-DD'));

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

VALUES (5, 'Eva White', 'HR Executive', 52000, 'HR', TO\_DATE('2019-08-05', 'YYYY-MM-DD'));

SELECT \* FROM Employees;



**-- Scenerio-1 --**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acct IN (

    SELECT AccountID, Balance

    FROM Accounts

    WHERE AccountType = 'Savings'

  ) LOOP

    UPDATE Accounts

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

        LastModified = SYSDATE

    WHERE AccountID = acct.AccountID;

  END LOOP;

  COMMIT;

END;

/

EXEC ProcessMonthlyInterest;

SELECT \* FROM Accounts;

**-- Scenerio-2 --**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

  dept\_name IN VARCHAR2,

  bonus\_pct IN NUMBER

) AS

BEGIN

  UPDATE Employees

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

  WHERE Department = dept\_name;

  COMMIT;

END;

/

EXEC UpdateEmployeeBonus('HR', 10);

SELECT \* FROM Employees;

**-- Scenerio-3 --**

CREATE OR REPLACE PROCEDURE TransferFunds(

  from\_acc IN NUMBER,

  to\_acc IN NUMBER,

  amt IN NUMBER

) AS

  from\_balance NUMBER;

  insufficient\_balance EXCEPTION;

BEGIN

  SELECT Balance INTO from\_balance FROM Accounts

  WHERE AccountID = from\_acc FOR UPDATE;

  IF from\_balance < amt THEN

    RAISE insufficient\_balance;

  END IF;

  UPDATE Accounts

  SET Balance = Balance - amt,

      LastModified = SYSDATE

  WHERE AccountID = from\_acc;

  UPDATE Accounts

  SET Balance = Balance + amt,

      LastModified = SYSDATE

  WHERE AccountID = to\_acc;

  COMMIT;

EXCEPTION

  WHEN insufficient\_balance THEN

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

    ROLLBACK;

  WHEN OTHERS THEN

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

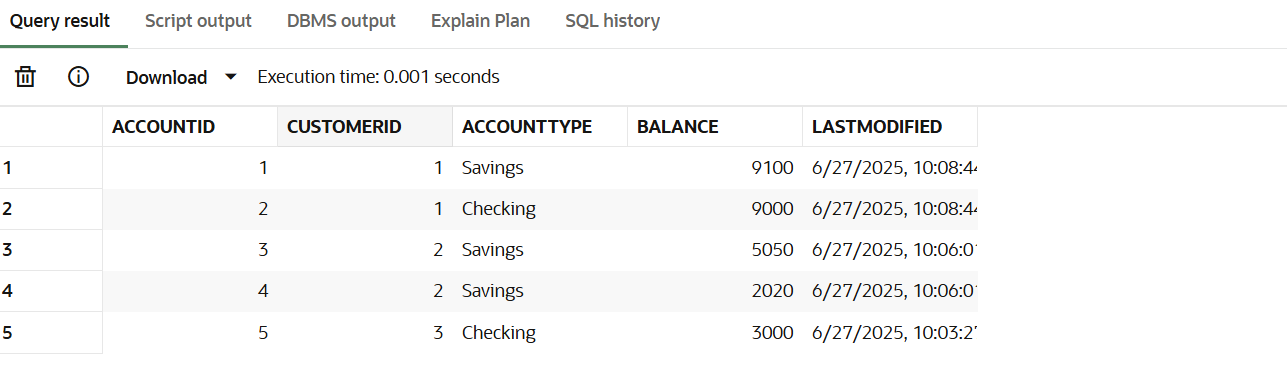
    ROLLBACK;

END;

/

EXEC TransferFunds(1, 2, 1000);

SELECT \* FROM Accounts;

****