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.

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.

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.

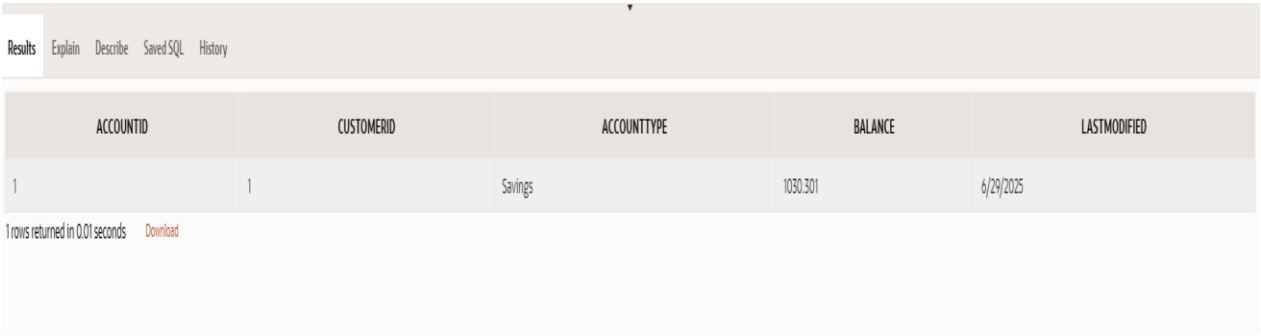
Scenario 1:

Code:

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

  COMMIT;
```

```
END;
BEGIN
   ProcessMonthlyInterest;
END;
SELECT AccountID, CustomerID, AccountType, Balance, LastModified
FROM Accounts
WHERE AccountType = 'Savings';
```

OUTPUT:



Results | Explain | Describe | Saved SQL | History

ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	1	Savings	1030.301	6/29/2025

1 rows returned in 0.01 seconds Download

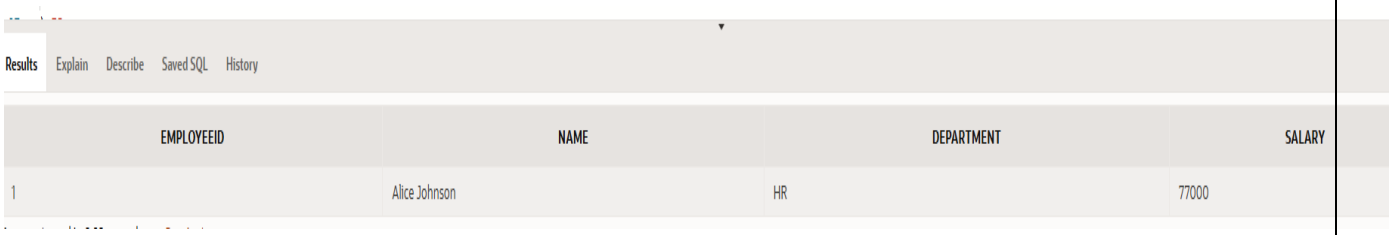
Scenario 2:

Code:

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

   COMMIT;
END;

BEGIN
   UpdateEmployeeBonus('HR', 10);
END;
```

```
SELECT EmployeeID, Name, Department, Salary
FROM Employees
WHERE Department = 'HR';
```

OUTPUT:



Results | Explain | Describe | Saved SQL | History

EMPLOYEEID	NAME	DEPARTMENT	SALARY
1	Alice Johnson	HR	77000

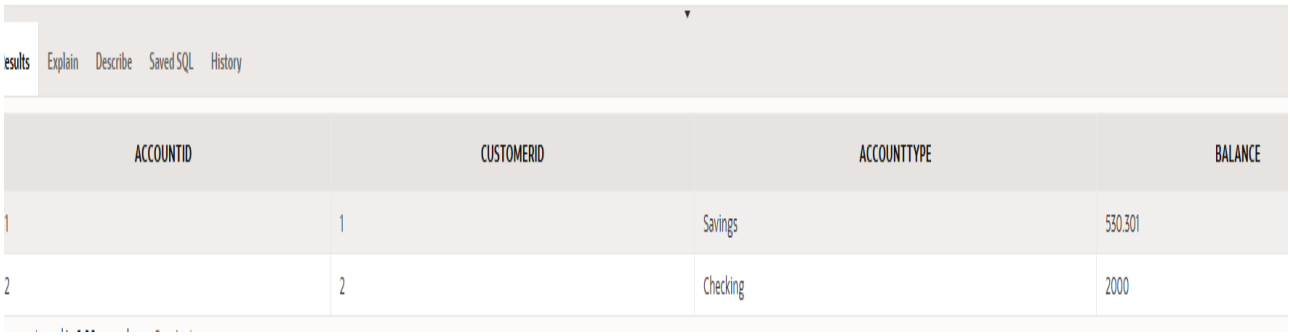
Scenario 3:

Code:

```
CREATE OR REPLACE PROCEDURE TransferFunds (
    from_account IN NUMBER,
    to_account IN NUMBER,
    amount IN NUMBER
) IS
    insufficient_balance EXCEPTION;
BEGIN
    -- Check if source has enough balance
    DECLARE
        src_balance NUMBER;
    BEGIN
        SELECT Balance INTO src_balance
        FROM Accounts
        WHERE AccountID = from_account;

        IF src_balance < amount THEN
            RAISE insufficient_balance;
        END IF;

        -- Deduct from source
        UPDATE Accounts
        SET Balance = Balance - amount,
```

```
            LastModified = SYSDATE
        WHERE AccountID = from_account;

        -- Add to destination
        UPDATE Accounts
        SET Balance = Balance + amount,
            LastModified = SYSDATE
        WHERE AccountID = to_account;

        COMMIT;
    END;

EXCEPTION
    WHEN insufficient_balance THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Transfer failed: Insufficient balance in source account.');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);
END;



BEGIN
    TransferFunds(1, 2, 500);
END;
SELECT AccountID, CustomerID, AccountType, Balance
FROM Accounts
WHERE AccountID IN (1, 2);
```

OUTPUT:



esults Explain Describe Saved SQL History

ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE
1	1	Savings	530.301
2	2	Checking	2000