

### 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;
```

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
```

```
    v_updated_rows NUMBER := 0;
```

```
BEGIN
```

```
    UPDATE Accounts
```

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

```
        LastModified = SYSDATE
```

```
    WHERE AccountType = 'Savings';
```

```
    v_updated_rows := SQL%ROWCOUNT;
```

```
    COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE(v_updated_rows || ' savings account(s) updated with 1% interest.');
```

```
    FOR rec IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP
```

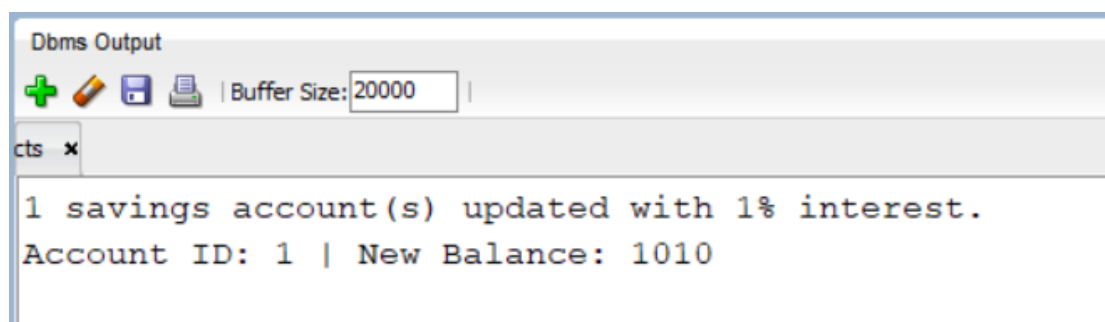
```
        DBMS_OUTPUT.PUT_LINE('Account ID: ' || rec.AccountID || ' | New Balance: ' || rec.Balance);
```

```
    END LOOP;
```

```
END;
```

```
/
```

```
EXEC ProcessMonthlyInterest;
```

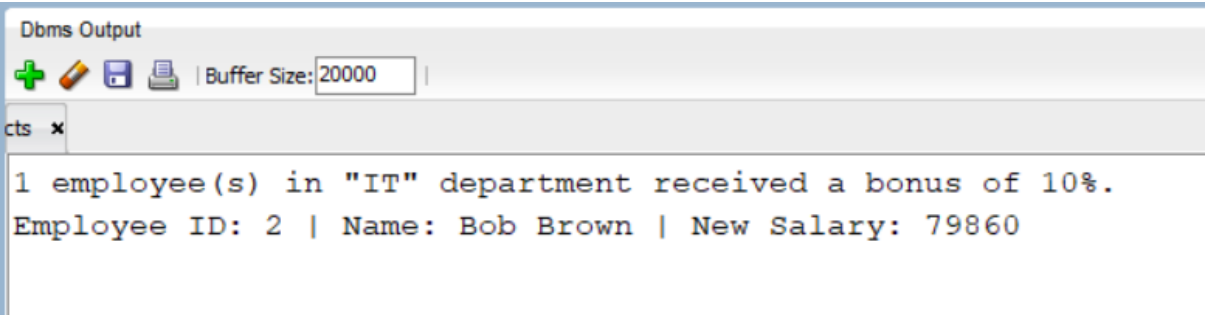


**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

**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_name IN VARCHAR2,  
    bonus_percent IN NUMBER  
) AS  
    v_updated_rows NUMBER := 0;  
BEGIN  
    UPDATE Employees  
    SET Salary = Salary + (Salary * bonus_percent / 100)  
    WHERE Department = dept_name;  
  
    v_updated_rows := SQL%ROWCOUNT;  
  
    COMMIT;  
  
    DBMS_OUTPUT.PUT_LINE(v_updated_rows || ' employee(s) in "' || dept_name || '"  
department received a bonus of ' || bonus_percent || '%');  
  
    FOR rec IN (SELECT EmployeeID, Name, Salary FROM Employees WHERE  
Department = dept_name) LOOP  
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.EmployeeID || ' | Name: ' ||  
rec.Name || ' | New Salary: ' || rec.Salary);  
    END LOOP;  
END;  
/  
  
EXEC UpdateEmployeeBonus('IT', 10);
```



The screenshot shows a 'Dbms Output' window with a buffer size of 20000. It displays the output of the stored procedure execution for the 'IT' department with a 10% bonus. The output consists of two lines: a summary line stating '1 employee(s) in "IT" department received a bonus of 10%.' and a detailed line for Employee ID 2, Bob Brown, showing his new salary as 79860.

```
1 employee(s) in "IT" department received a bonus of 10%.  
Employee ID: 2 | Name: Bob Brown | New Salary: 79860
```

**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:**

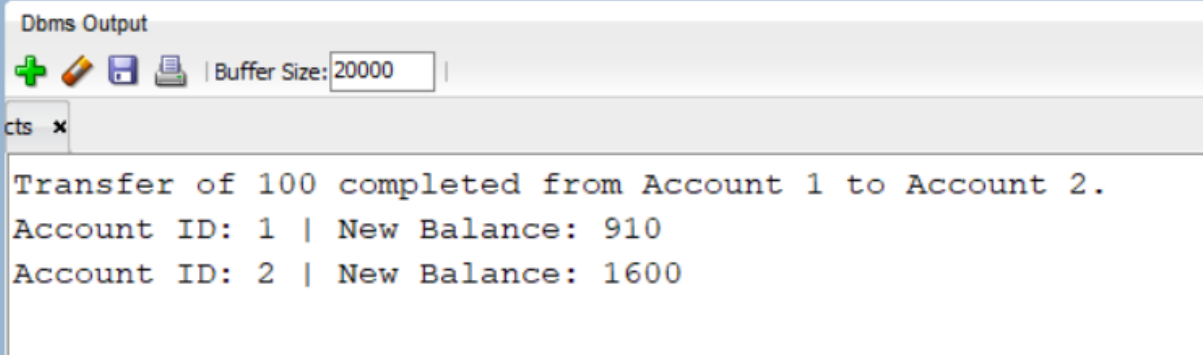
```
CREATE OR REPLACE PROCEDURE TransferFunds (  
    from_account_id IN NUMBER,  
    to_account_id IN NUMBER,  
    amount IN NUMBER  
) AS  
    from_balance NUMBER;  
BEGIN  
    SELECT Balance INTO from_balance  
    FROM Accounts  
    WHERE AccountID = from_account_id  
    FOR UPDATE;  
  
    IF from_balance < amount THEN  
        DBMS_OUTPUT.PUT_LINE('Transfer failed: Insufficient balance in Account ' ||  
from_account_id);  
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance.');
```

```
    END IF;  
  
    -- Deduct amount  
    UPDATE Accounts  
    SET Balance = Balance - amount,  
        LastModified = SYSDATE  
    WHERE AccountID = from_account_id;  
  
    -- Add amount  
    UPDATE Accounts  
    SET Balance = Balance + amount,  
        LastModified = SYSDATE  
    WHERE AccountID = to_account_id;  
  
    COMMIT;  
  
    DBMS_OUTPUT.PUT_LINE('Transfer of ' || amount || ' completed from Account ' ||  
from_account_id || ' to Account ' || to_account_id || '.');
```

```
    -- Show new balances
```

```
FOR rec IN (SELECT AccountID, Balance FROM Accounts WHERE AccountID IN
(from_account_id, to_account_id)) LOOP
    DBMS_OUTPUT.PUT_LINE('Account ID: ' || rec.AccountID || ' | New Balance: ' ||
rec.Balance);
END LOOP;
END;
/

EXEC TransferFunds(1, 2, 100);
```



The screenshot shows a 'Dbms Output' window with a toolbar containing icons for adding, editing, saving, and printing, along with a 'Buffer Size' field set to 20000. Below the toolbar, a tab labeled 'cts x' is visible. The main area of the window displays the following text:

```
Transfer of 100 completed from Account 1 to Account 2.
Account ID: 1 | New Balance: 910
Account ID: 2 | New Balance: 1600
```