

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:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

    FOR rec IN (SELECT account_id, balance FROM accounts WHERE account_type = 'savings') LOOP

        UPDATE accounts

        SET balance = balance + (balance * 0.01)

        WHERE account_id = rec.account_id;

    END LOOP;

    COMMIT;

END;

/

BEGIN

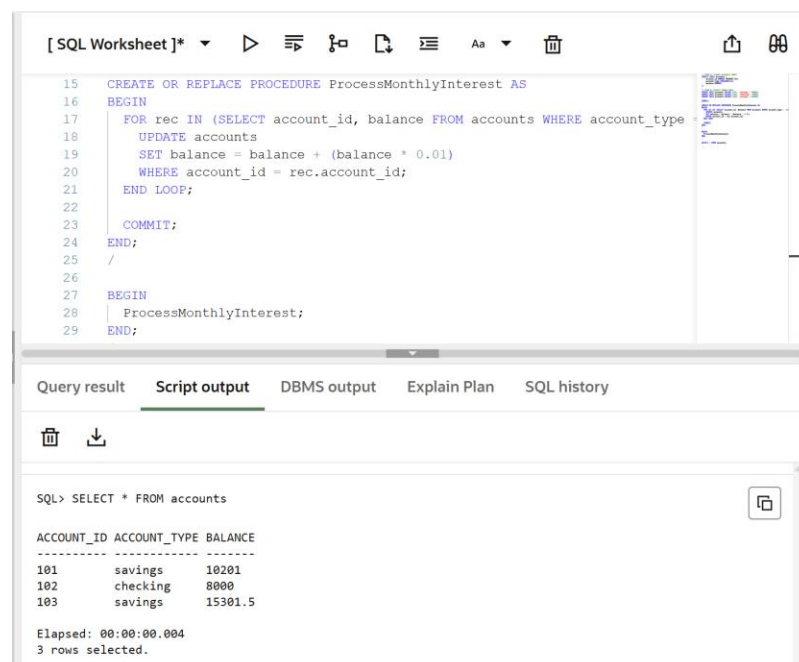
    ProcessMonthlyInterest;

END;

/

SELECT * FROM accounts;
```

Output:



The screenshot shows a SQL IDE interface with a script editor and a results pane. The script editor contains the following SQL code:

```
15 CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
16 BEGIN
17     FOR rec IN (SELECT account_id, balance FROM accounts WHERE account_type = 'savings') LOOP
18         UPDATE accounts
19         SET balance = balance + (balance * 0.01)
20         WHERE account_id = rec.account_id;
21     END LOOP;
22     COMMIT;
23 END;
24 /
25
26 BEGIN
27     ProcessMonthlyInterest;
28 END;
29 /
```

The results pane shows the output of the query `SELECT * FROM accounts;`. The output is a table with three columns: `ACCOUNT_ID`, `ACCOUNT_TYPE`, and `BALANCE`. The data is as follows:

ACCOUNT_ID	ACCOUNT_TYPE	BALANCE
101	savings	10201
102	checking	8000
103	savings	15301.5

Below the table, the execution time is shown as `Elapsed: 00:00:00.004` and the number of rows selected is `3 rows selected.`

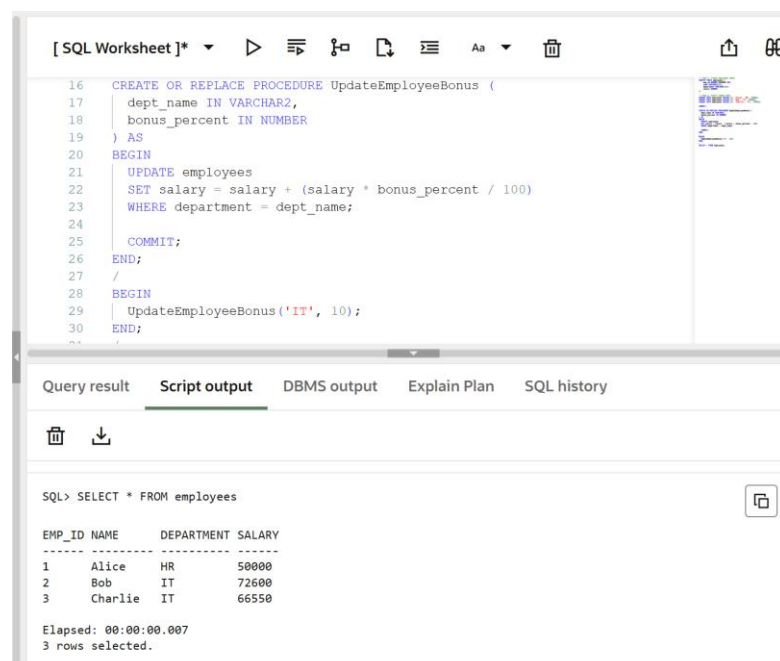
Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

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:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (  
    dept_name IN VARCHAR2,  
    bonus_percent IN NUMBER  
) AS  
BEGIN  
    UPDATE employees  
    SET salary = salary + (salary * bonus_percent / 100)  
    WHERE department = dept_name;  
    COMMIT;  
END;  
  
/  
  
BEGIN  
    UpdateEmployeeBonus('IT', 10);  
END;  
  
/  
  
SELECT * FROM employees;
```

Output:



The screenshot shows an SQL worksheet with the following content:

```
16 CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (  
17     dept_name IN VARCHAR2,  
18     bonus_percent IN NUMBER  
19 ) AS  
20 BEGIN  
21     UPDATE employees  
22     SET salary = salary + (salary * bonus_percent / 100)  
23     WHERE department = dept_name;  
24     COMMIT;  
25 END;  
26 /  
27  
28 BEGIN  
29     UpdateEmployeeBonus('IT', 10);  
30 END;  
31 /  
32  
33 SELECT * FROM employees;
```

The output tab shows the following results:

EMP_ID	NAME	DEPARTMENT	SALARY
1	Alice	HR	50000
2	Bob	IT	72600
3	Charlie	IT	66550

Elapsed: 00:00:00.007
3 rows selected.

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:

```
CREATE OR REPLACE PROCEDURE TransferFunds (  
    from_account IN NUMBER,  
    to_account IN NUMBER,  
    amount IN NUMBER  
) AS  
    insufficient_balance EXCEPTION;  
    v_balance NUMBER;  
BEGIN  
    SELECT balance INTO v_balance FROM accounts WHERE account_id = from_account;  
  
    IF v_balance < amount THEN  
        RAISE insufficient_balance;  
    END IF;  
  
    UPDATE accounts  
    SET balance = balance - amount  
    WHERE account_id = from_account;  
  
    UPDATE accounts  
    SET balance = balance + amount  
    WHERE account_id = to_account;  
  
    COMMIT;  
EXCEPTION  
    WHEN insufficient_balance THEN  
        DBMS_OUTPUT.PUT_LINE('Transfer failed: Insufficient funds.');    WHEN NO_DATA_FOUND THEN  
        DBMS_OUTPUT.PUT_LINE('Transfer failed: Account not found.');    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);  
END;  
  
/  
  
BEGIN  
    TransferFunds(101, 102, 2000);
```

END;

/

SELECT * FROM accounts;

Output:


Query result


Script output

DBMS output

Explain Plan

SQL history





SQL> CREATE OR REPLACE PROCEDURE TransferFunds (
from_account IN NUMBER,
to_account IN NUMBER,
amount IN NUMBER...
[Show more...](#)

Procedure TRANSFERFUNDS compiled

Elapsed: 00:00:00.015

SQL> BEGIN
TransferFunds(101, 102, 2000);
END;

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

SQL> SELECT * FROM accounts

ACCOUNT_ID	ACCOUNT_TYPE	BALANCE
101	savings	4201
102	checking	14000
103	savings	15301.5

Elapsed: 00:00:00.004
3 rows selected.