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 is
   bal accounts.balance%type;
begin
   for acc in (
      select accountid,
             balance
        from accounts
   ) loop
      bal := acc.balance;
      update accounts
         set
         balance = bal * 1.01
       where accountid = acc.accountid;

      dbms_output.put_line('ACCOUNT ID: '
                           || acc.accountid
                           || ', OLD BALANCE: '
                           || bal
                           || ', NEW BALANCE: '
                           ||(bal * 1.01));
   end loop;
   commit;
end;
/

EXEC PROCESSMONTHLYINTEREST;
```

OUTPUT:

```
SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
         bal ACCOUNTS.BALANCE%TYPE;
     BEGIN
         FOR acc IN (...
Show more...
```

```
Procedure PROCESSMONTHLYINTEREST compiled

Elapsed: 00:00:00.024
```

```
SQL> EXEC PROCESSMONTHLYINTEREST


ACCOUNT ID: 1, OLD BALANCE: 1000, NEW BALANCE: 1010
ACCOUNT ID: 2, OLD BALANCE: 1500, NEW BALANCE: 1515


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.010
```

Initial Accounts table:

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	1	1	Savings	1000	6/26/2025, 10:18:03 AM
2	2	2	Checking	1500	6/26/2025, 10:18:03 AM

Updated Accounts Table:

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	1	1	Savings	1010	6/26/2025, 10:18:03 AM
2	2	2	Checking	1515	6/26/2025, 10:18:03 AM

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 (
   bonusPercentage IN NUMBER,
   deptName IN VARCHAR2
) IS
   bonus employees.salary%TYPE;
   sal   employees.salary%TYPE;
BEGIN
   FOR emp IN (
      SELECT employeeid, salary
      FROM employees
      WHERE department = deptName
   ) LOOP
      sal := emp.salary;
      bonus := sal * (100 + bonusPercentage) / 100;

      UPDATE employees
      SET salary = bonus
      WHERE employeeid = emp.employeeid;

      DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employeeid ||
                           ', Old salary: ' || sal ||
                           ', New salary: ' || bonus);
   END LOOP;

   COMMIT;
END;
/

EXEC UPDATEEMPLOYEEBONUS(5, 'HR');
```

OUTPUT:

```
SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
        bonusPercentage IN NUMBER,
        deptName IN VARCHAR2
     ) IS...
Show more...


Procedure UPDATEEMPLOYEEBONUS compiled

Elapsed: 00:00:00.019

SQL> Exec UPDATEEMPLOYEEBONUS(5, 'HR')


Employee ID: 1, Old salary: 70000, New salary: 73500


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009
```

Initial Employees Table:

	EMPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	1	Alice Johnson	Manager	70000	HR	6/15/2015, 12:00:00 AM
2	2	Bob Brown	Developer	60000	IT	3/20/2017, 12:00:00 AM

Updated Employees Table:

Download Execution time: 0.001 seconds

	EMPLOYEEID	NAME	POSITION	SALARY	DEPARTMENT	HIREDATE
1	1	Alice Johnson	Manager	73500	HR	6/15/2015, 12:00:00 AM
2	2	Bob Brown	Developer	60000	IT	3/20/2017, 12:00:00 AM

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 (
   sender   in accounts.accountid%type,
   receiver in accounts.accountid%type,
   amount   in number
) is
   sender_balance   accounts.balance%type;
   receiver_balance accounts.balance%type;
begin
   select balance
     into sender_balance
     from accounts
    where accountid = sender;

   select balance
     into receiver_balance
     from accounts
    where accountid = receiver;

   if sender_balance < amount then
      dbms_output.put_line('Insufficient balance.');
   elsif amount <= 0 then
      dbms_output.put_line('Amount cannot be zero or less.');
   else
      update accounts
         set
         balance = balance - amount
       where accountid = sender;

      update accounts
         set
         balance = balance + amount
       where accountid = receiver;

      commit;
      dbms_output.put_line('Transfer successful.');
   end if;

end;
```

OUTPUT:

```
SQL> EXEC TRANSFERFUNDS(1,2,200)


Tranfer Successful.


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.016
```

Initial Accounts Table:

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	1	1	Savings	1010	6/26/2025, 10:18:03 AM
2	2	2	Checking	1515	6/26/2025, 10:18:03 AM

Modified Accounts Table:

	ACCOUNTID	CUSTOMERID	ACCOUNTTYPE	BALANCE	LASTMODIFIED
1	1	1	Savings	810	6/26/2025, 10:18:0
2	2	2	Checking	1715	6/26/2025, 10:18:0