# PL/SQL Mandatory Hands On Exercise 3

QUESTION:

Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.

o 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.

o 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.

o 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 (Scenario 1):

**CREATE TABLE savings\_accounts (**

**account\_id NUMBER PRIMARY KEY,**

**customer\_id NUMBER,**

**balance NUMBER(12, 2)**

**);**

**INSERT INTO savings\_accounts VALUES (101, 1, 10000);**

**INSERT INTO savings\_accounts VALUES (102, 2, 15000);**

**INSERT INTO savings\_accounts VALUES (103, 3, 12000);**

**INSERT INTO savings\_accounts VALUES (104, 4, 8000);**

**INSERT INTO savings\_accounts VALUES (105, 5, 9500);**

**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS**

**BEGIN**

**FOR rec IN (SELECT account\_id, balance FROM savings\_accounts) LOOP**

**UPDATE savings\_accounts**

**SET balance = balance + (balance \* 0.01)**

**WHERE account\_id = rec.account\_id;**

**END LOOP;**

**COMMIT;**

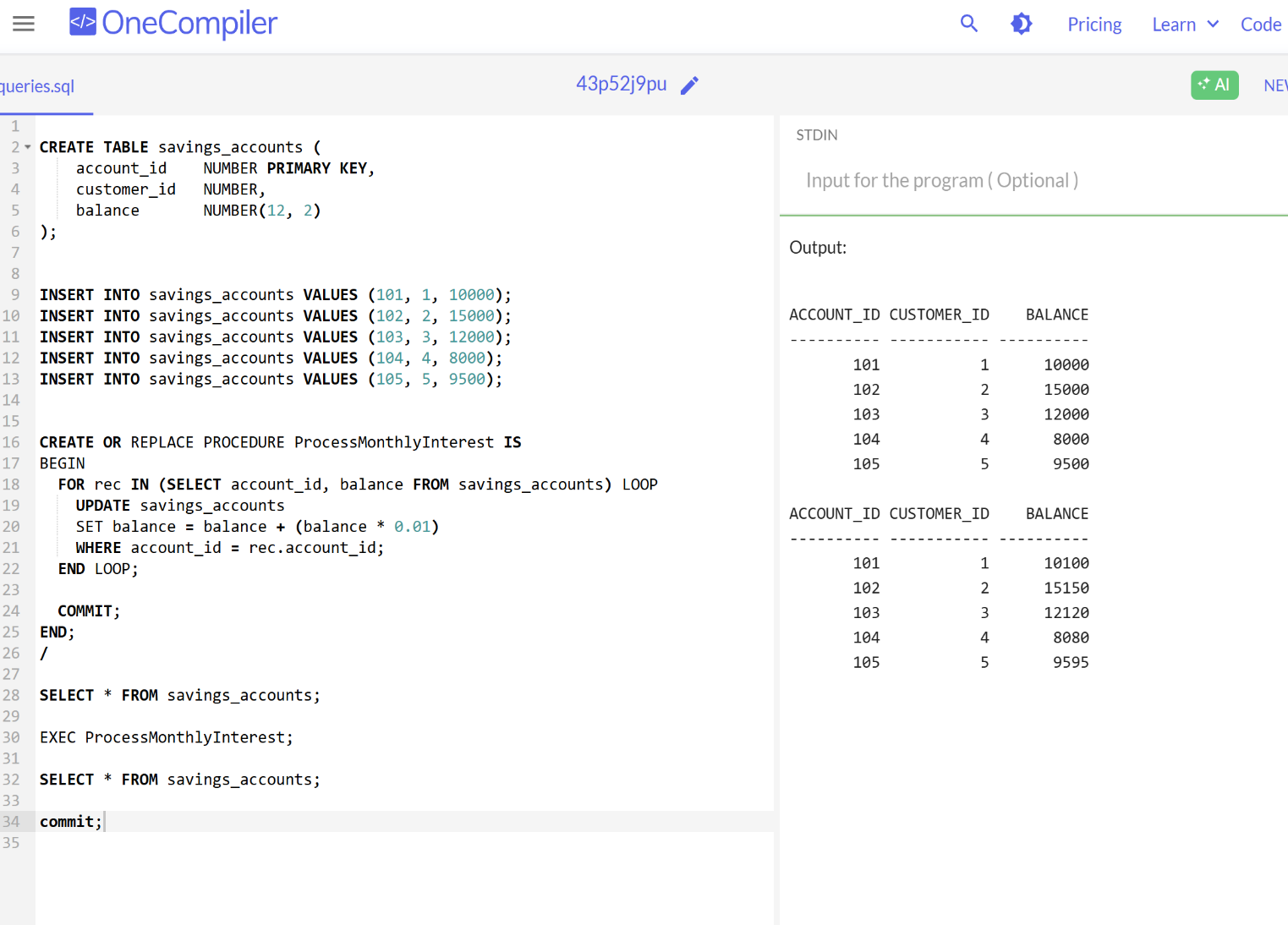
**END;**

**/**

**SELECT \* FROM savings\_accounts;**

**EXEC ProcessMonthlyInterest;**

**SELECT \* FROM savings\_accounts;**

**commit;**OUTPUT :

CODE(Scenario 2):

**CREATE TABLE employees (**

**emp\_id NUMBER PRIMARY KEY,**

**name VARCHAR2(30),**

**department VARCHAR2(30),**

**salary NUMBER(10, 2)**

**);**

**INSERT INTO employees VALUES (1, 'Alice', 'Sales', 50000);**

**INSERT INTO employees VALUES (2, 'Bob', 'IT', 60000);**

**INSERT INTO employees VALUES (3, 'Charlie', 'HR', 45000);**

**INSERT INTO employees VALUES (4, 'Diana', 'Sales', 55000);**

**INSERT INTO employees VALUES (5, 'Eve', 'IT', 62000);**

**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(**

**p\_department IN VARCHAR2,**

**p\_bonus\_percent IN NUMBER**

**) IS**

**BEGIN**

**UPDATE employees**

**SET salary = salary + (salary \* p\_bonus\_percent / 100)**

**WHERE department = p\_department;**

**COMMIT;**

**END;**

**/**

**SET LINESIZE 150;**

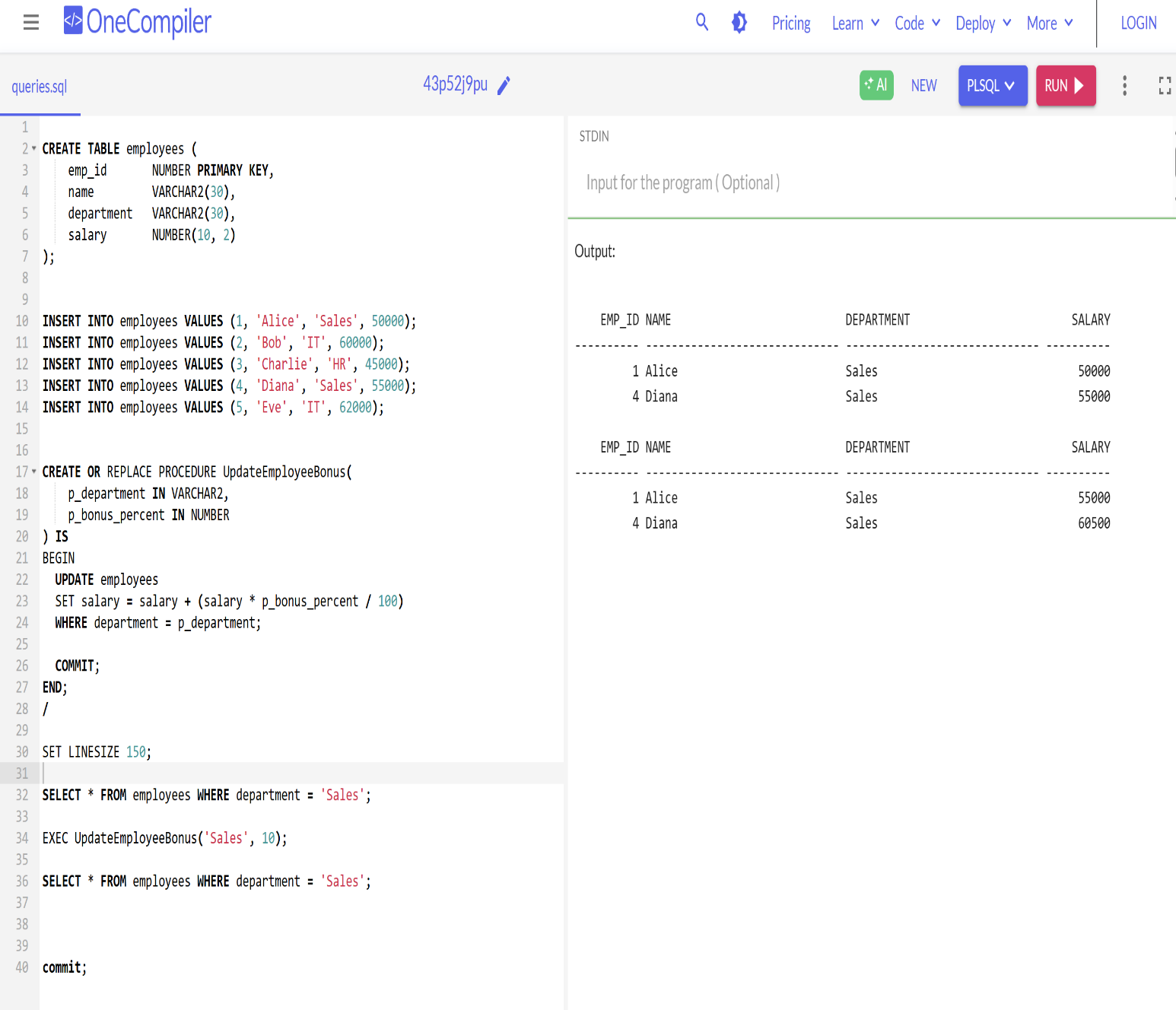
**SELECT \* FROM employees WHERE department = 'Sales';**

**EXEC UpdateEmployeeBonus('Sales', 10);**

**SELECT \* FROM employees WHERE department = 'Sales';**

**commit;**

OUTPUT:



CODE (SCENARIO 3):

**CREATE TABLE accounts (**

**account\_id NUMBER PRIMARY KEY,**

**customer\_id NUMBER,**

**balance NUMBER(12, 2)**

**);**

**INSERT INTO accounts VALUES (201, 1, 8000);**

**INSERT INTO accounts VALUES (202, 2, 10000);**

**INSERT INTO accounts VALUES (203, 3, 5000);**

**INSERT INTO accounts VALUES (204, 4, 12000);**

**INSERT INTO accounts VALUES (205, 5, 7000);**

**CREATE OR REPLACE PROCEDURE TransferFunds(**

**p\_from\_account IN NUMBER,**

**p\_to\_account IN NUMBER,**

**p\_amount IN NUMBER**

**) IS**

**v\_balance NUMBER;**

**BEGIN**

**SELECT balance INTO v\_balance FROM accounts WHERE account\_id = p\_from\_account;**

**IF v\_balance < p\_amount THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');**

**ELSE**

**UPDATE accounts**

**SET balance = balance - p\_amount**

**WHERE account\_id = p\_from\_account;**

**UPDATE accounts**

**SET balance = balance + p\_amount**

**WHERE account\_id = p\_to\_account;**

**COMMIT;**

**END IF;**

**END;**

**/**

**SELECT \* FROM accounts WHERE account\_id IN (201, 202);**

**EXEC TransferFunds(201, 202, 2000);**

**SELECT \* FROM accounts WHERE account\_id IN (201, 202);**

**commit;**

OUTPUT:

