Digital Nurture 4.0 – Deep Skilling Program

Java Full Stack Engineer (FSE)

Week 2 – Mandatory Exercises Submission

Name: Adishree Devoor

GitHub Profile: https://github.com/Adishree16

Email: [adishreedevoor18@gmail.com](mailto:adishreedevoor18@gmail.com)

Skills Covered:

* PL/SQL Programming

**PLSQL Exercises**

**Exercise 1: Control Structures**

**Program:**

-- Create customers table

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

isVIP VARCHAR2(5)

);

-- Create loans table

CREATE TABLE loans (

loan\_id NUMBER PRIMARY KEY,

customer\_id NUMBER REFERENCES customers(customer\_id),

interest\_rate NUMBER,

due\_date DATE

);

-- Insert sample data

INSERT INTO customers VALUES (1, 'John Doe', 65, 15000, 'FALSE');

INSERT INTO customers VALUES (2, 'Jane Smith', 45, 8000, 'FALSE');

INSERT INTO loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO loans VALUES (102, 2, 8.0, SYSDATE + 40);

COMMIT;

Scenario 1: Discount for customers above 60

**Program:**

BEGIN

FOR cust IN (SELECT customer\_id, age FROM customers) LOOP

IF cust.age > 60 THEN

UPDATE loans

SET interest\_rate = interest\_rate \* 0.99

WHERE customer\_id = cust.customer\_id;

END IF;

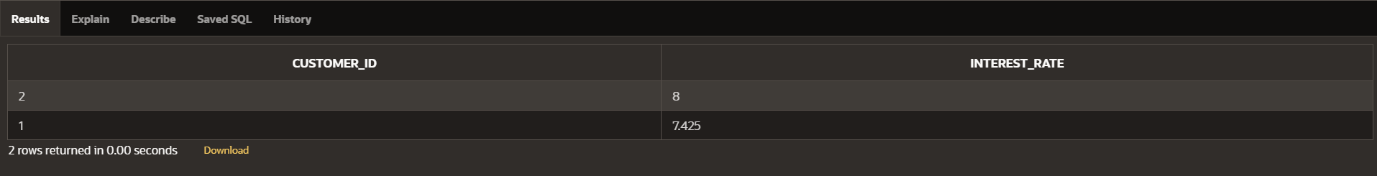
END LOOP;

COMMIT;

END;

/

**Output:**



Scenario 2: Promote VIP if balance > $10,000

**Program:**

BEGIN

FOR cust IN (SELECT customer\_id, balance FROM customers) LOOP

IF cust.balance > 10000 THEN

UPDATE customers

SET isVIP = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END IF;

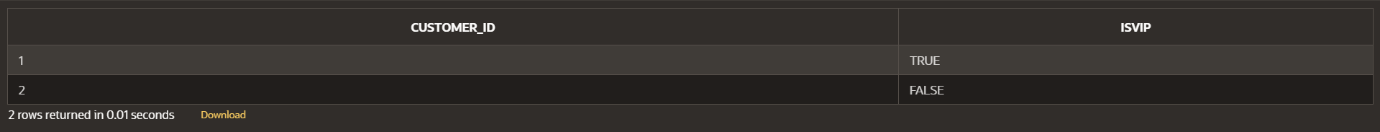
END LOOP;

COMMIT;

END;

/

**Output:**



Scenario 3: Reminder for loans due in next 30 days

**Program:**

BEGIN

FOR loan IN (SELECT customer\_id, due\_date FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ' || loan.customer\_id ||

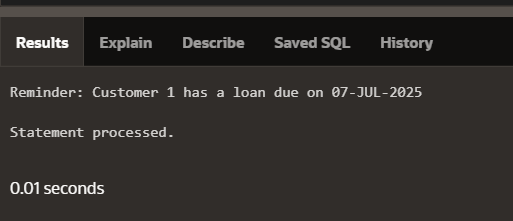
' has a loan due on ' || TO\_CHAR(loan.due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

/

**Output:**



**Exercise 3: Stored Procedures**

**Program:**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Salary NUMBER(10,2),

DepartmentID NUMBER

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Salary NUMBER(10,2),

DepartmentID NUMBER

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(100),

Balance NUMBER(12,2)

);

INSERT INTO SavingsAccount (AccountID, CustomerName, Balance) VALUES (101, 'Alice', 10000);

INSERT INTO SavingsAccount (AccountID, CustomerName, Balance) VALUES (102, 'Bob', 15000);

INSERT INTO SavingsAccount (AccountID, CustomerName, Balance) VALUES (103, 'Charlie', 20000);

INSERT INTO Employees (EmployeeID, Name, Salary, DepartmentID) VALUES (201, 'David', 50000, 10);

INSERT INTO Employees (EmployeeID, Name, Salary, DepartmentID) VALUES (202, 'Emma', 55000, 10);

INSERT INTO Employees (EmployeeID, Name, Salary, DepartmentID) VALUES (203, 'Frank', 60000, 20);

INSERT INTO Accounts (AccountID, CustomerName, Balance) VALUES (301, 'Grace', 20000);

INSERT INTO Accounts (AccountID, CustomerName, Balance) VALUES (302, 'Hannah', 10000);  
COMMIT;

Scenario 1: ProcessMonthlyInterest

**Program:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

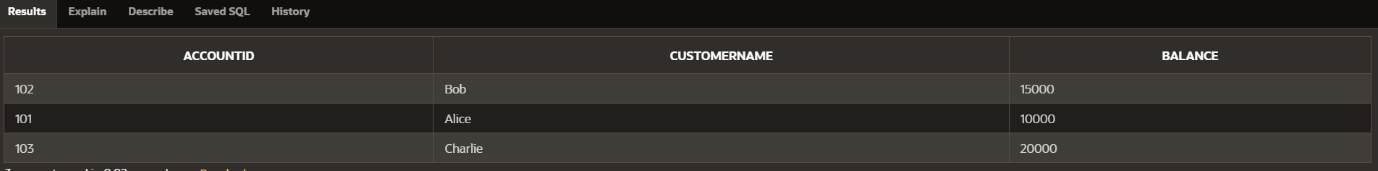
BEGIN

UPDATE SavingsAccount

SET Balance = Balance + (Balance \* 0.01);

END;

**Output:**



Scenario 2: UpdateEmployeeBonus

**Program:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) IS

BEGIN

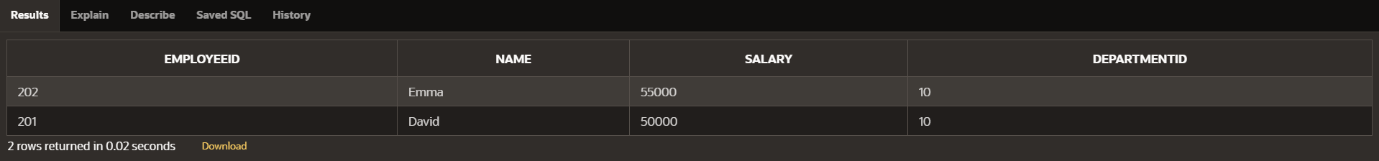
UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE DepartmentID = dept\_id;

END;

**Output:**



Scenario 3: TransferFunds

**Program:**

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) IS

insufficient\_balance EXCEPTION;

BEGIN

-- Check for sufficient balance

FOR acc IN (SELECT Balance FROM Accounts WHERE AccountID = from\_acc) LOOP

IF acc.Balance < amount THEN

RAISE insufficient\_balance;

END IF;

END LOOP;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_acc;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_acc;

EXCEPTION

WHEN insufficient\_balance THEN

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

END;

**Output:**

