**Exercise 3: Stored Procedures**

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

**Question:** Write a stored procedure **Process Monthly Interest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Step 1: Create Customers Table**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

**Step 2: Create Accounts Table**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

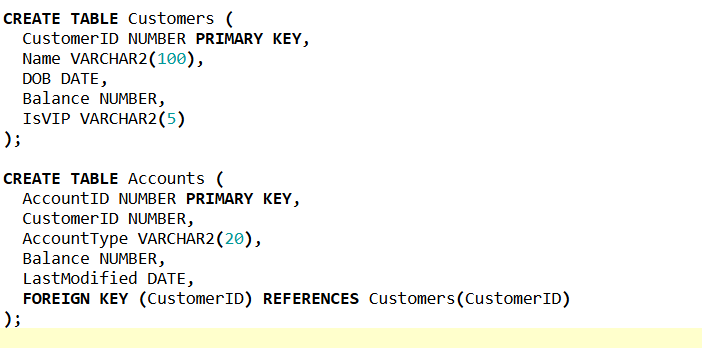
CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

); 

**Step 3: Insert Sample Data**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, IsVIP)

VALUES (1, 'John Doe', TO\_DATE('1980-05-10', 'YYYY-MM-DD'), 5000, 'FALSE');

INSERT INTO Customers (CustomerID, Name, DOB, Balance, IsVIP)

VALUES (2, 'Jane Smith', TO\_DATE('1990-08-22', 'YYYY-MM-DD'), 8000, 'FALSE');

INSERT INTO Customers (CustomerID, Name, DOB, Balance, IsVIP)

VALUES (3, 'Arun Vijay', TO\_DATE('1995-01-05', 'YYYY-MM-DD'), 12000, 'FALSE');

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 10000, SYSDATE);

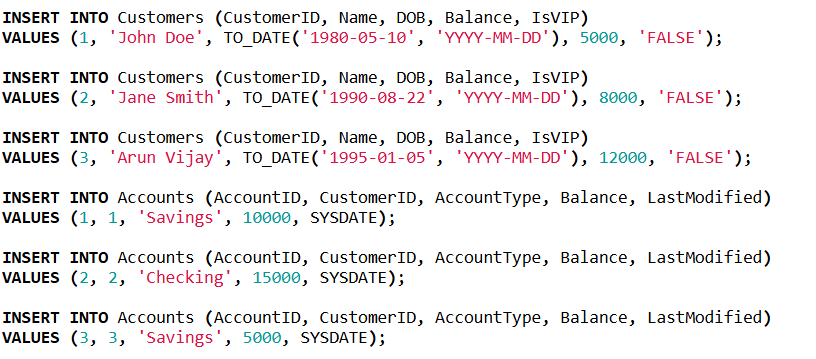
INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 15000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (3, 3, 'Savings', 5000, SYSDATE);

COMMIT;



**Step 4: Create the Procedure**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc\_rec IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (acc\_rec.Balance \* 0.01),

LastModified = SYSDATE

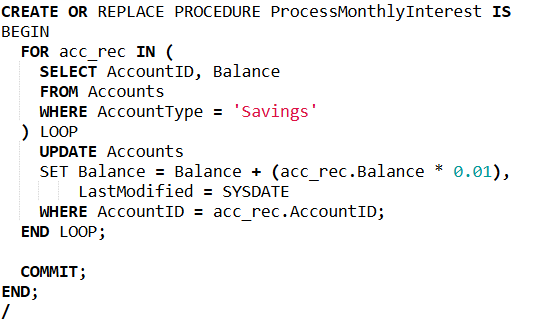
WHERE AccountID = acc\_rec.AccountID;

END LOOP;

COMMIT;

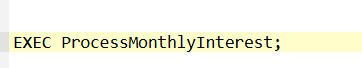
END;

/



**Step 5: Run the Procedure**

EXEC ProcessMonthlyInterest;

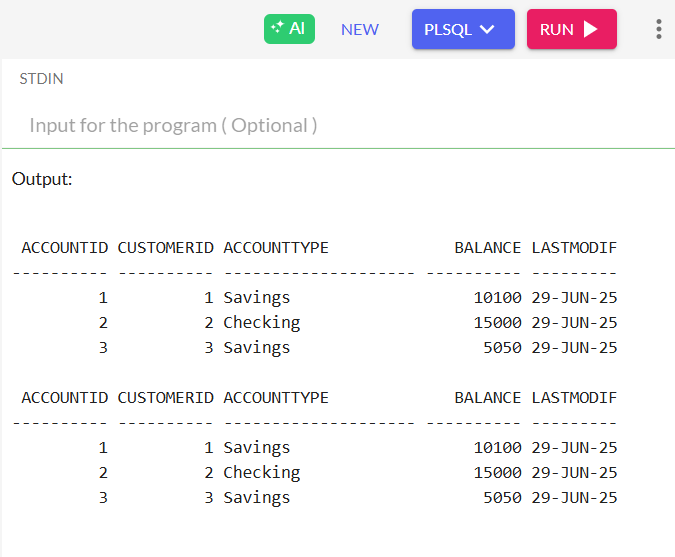


**Step 6: View the Updated Accounts**

SELECT \* FROM Accounts;



**Excepted output**

****

**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.

**Step 1: Create the Employees Table**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

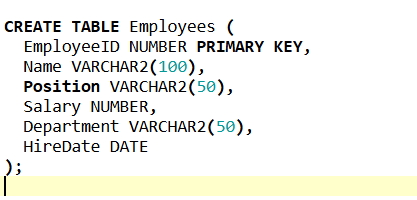
Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);



**Step 2: Insert Sample Employee Data**

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

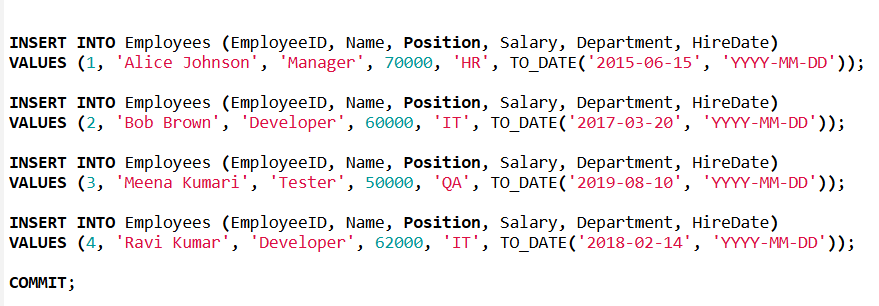
INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (3, 'Meena Kumari', 'Tester', 50000, 'QA', TO\_DATE('2019-08-10', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (4, 'Ravi Kumar', 'Developer', 62000, 'IT', TO\_DATE('2018-02-14', 'YYYY-MM-DD'));

COMMIT;



**Step 3: Enable Output for Console Logging**

SET SERVEROUTPUT ON;

**Step 4: Create the UpdateEmployeeBonus Stored Procedure**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp\_rec IN (

SELECT EmployeeID, Name, Salary

FROM Employees

WHERE Department = dept\_name

) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp\_rec.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to ' || emp\_rec.Name ||

': Old Salary = Rs' || emp\_rec.Salary ||

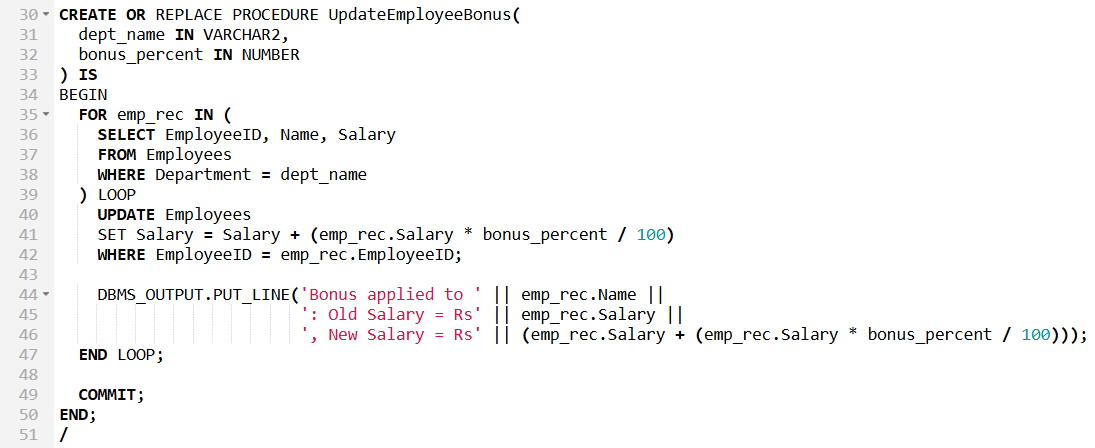
', New Salary = Rs' || (emp\_rec.Salary + (emp\_rec.Salary \* bonus\_percent / 100)));

END LOOP;

COMMIT;

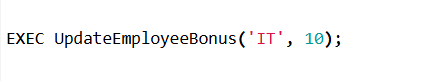
END;

/

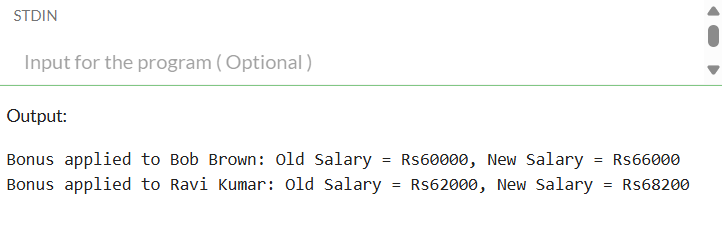


**Step 5: Execute the Procedure for a Department**

EXEC UpdateEmployeeBonus('IT', 10);

****

**Output:**

****

**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.

**Step 1: Create Customers Table**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

**Step 2: Create Accounts Table**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

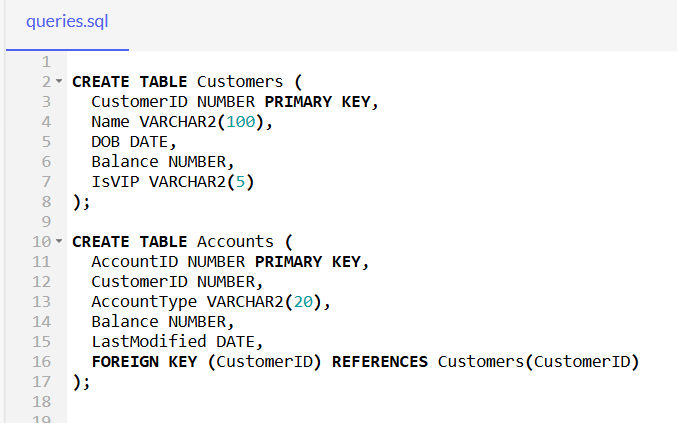
CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

); 

**Step 3: Insert Sample Data**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, IsVIP)

VALUES (1, 'John Doe', TO\_DATE('1980-05-10', 'YYYY-MM-DD'), 5000, 'FALSE');

INSERT INTO Customers (CustomerID, Name, DOB, Balance, IsVIP)

VALUES (2, 'Jane Smith', TO\_DATE('1990-08-22', 'YYYY-MM-DD'), 8000, 'FALSE');

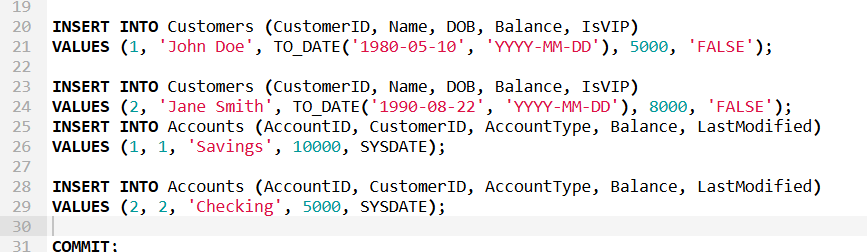
INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 10000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 5000, SYSDATE);

COMMIT;



**Step 4: Create TransferFunds Procedure**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE TransferFunds (

source\_account\_id IN NUMBER,

destination\_account\_id IN NUMBER,

amount IN NUMBER

) IS

v\_source\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_source\_balance

FROM Accounts

WHERE AccountID = source\_account\_id;

IF v\_source\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance.');

ELSE

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = source\_account\_id;

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = destination\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: Rs' || amount ||

' transferred from Account ' || source\_account\_id ||

' to Account ' || destination\_account\_id);

COMMIT;

END IF;

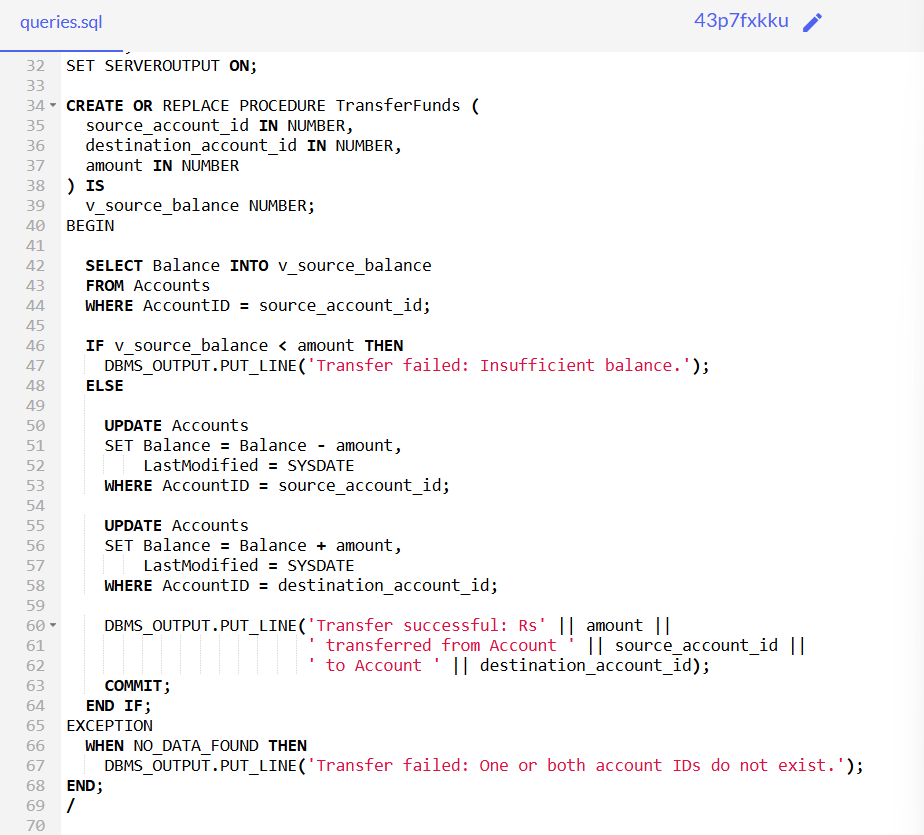
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One orboth account IDs do not exist.');

END;

/

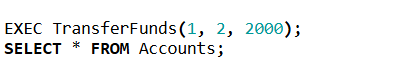


**Step 5: Run the Procedure**

EXEC TransferFunds(1, 2, 2000);

**Step 6: Check Final Balances**

SELECT \* FROM Accounts;



**Output:**

