**Week 2**

**PLSQL\_Exercises**

**SOLUTION**

***Schema Created:***

-- Create Tables

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

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

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

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

VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 12000, SYSDATE);

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

VALUES (1, 1, 'Savings', 1000, SYSDATE);

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

VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

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'));

SELECT \* FROM Customers;

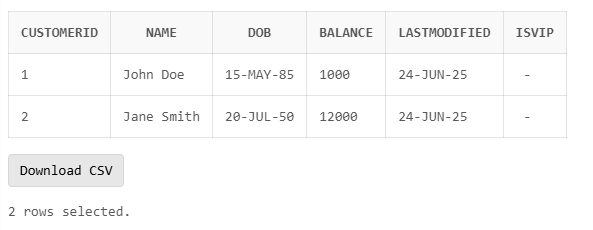
SELECT \* FROM Accounts;

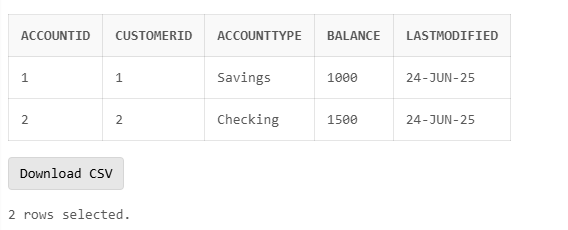
SELECT \* FROM Transactions;

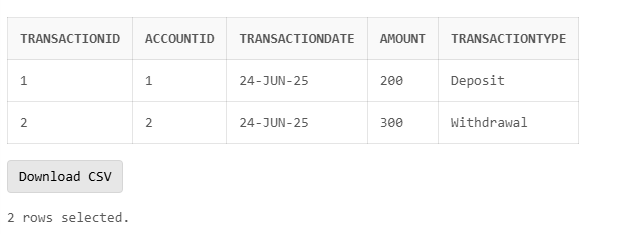
SELECT \* FROM Loans;

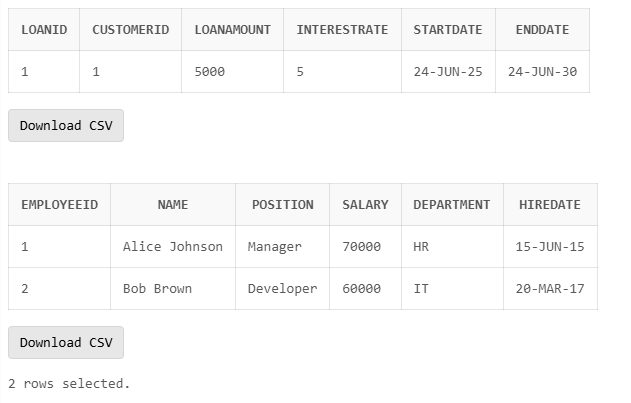
SELECT \* FROM Employees;

**OUTPUT**

****

****

****

****

**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* + **Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**CODE**

BEGIN

FOR cust IN (SELECT CustomerID, Name, DOB FROM Customers) LOOP

IF MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(

'Discount applied: ' || cust.Name ||

' (CustomerID: ' || cust.CustomerID ||

') - Age: ' || TRUNC(MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12)

);

END IF;

END LOOP;

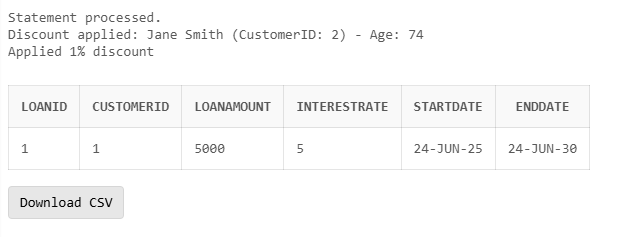
DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount');

END;

/

SELECT \* FROM Loans;

**OUTPUT**

****

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + **Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**CODE**

BEGIN

FOR cust IN (SELECT CustomerID, Name, Balance FROM Customers) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(

cust.Name || ' (ID: ' || cust.CustomerID || ') promoted to VIP - Balance: $' || cust.Balance

);

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE(

cust.Name || ' (ID: ' || cust.CustomerID || ') is not VIP - Balance: $' || cust.Balance

);

END IF;

END LOOP;

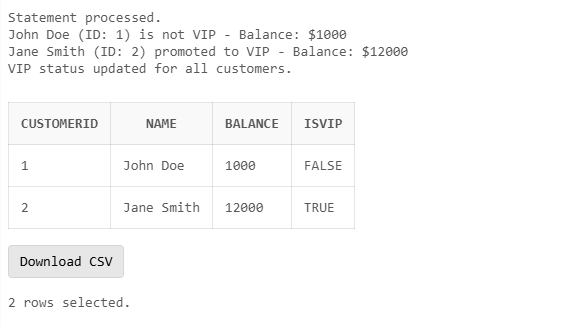
DBMS\_OUTPUT.PUT\_LINE('VIP status updated for all customers.');

END;

/

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

**OUTPUT**

****

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + **Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**CODE**

BEGIN

FOR loan IN (

SELECT l.LoanID, l.CustomerID, l.EndDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Dear ' || loan.Name ||

', your loan (ID: ' || loan.LoanID ||

') is due on ' || TO\_CHAR(loan.EndDate, 'DD-MON-YYYY')

);

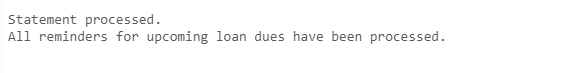
END LOOP;

DBMS\_OUTPUT.PUT\_LINE('All reminders for upcoming loan dues have been processed.');

END;

/

**OUTPUT**

****

**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 acc IN (

SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE(

'Interest added to AccountID ' || acc.AccountID ||

' | New Balance: ' || (acc.Balance + acc.Balance \* 0.01)

);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Monthly Interest updated for all savings accounts.');

END;

/

BEGIN

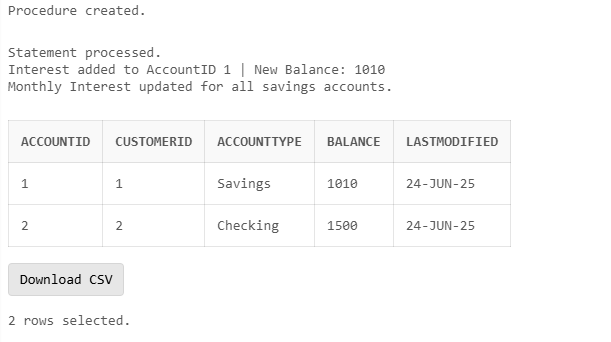
ProcessMonthlyInterest;

END;

/

SELECT \* FROM Accounts;

**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.

**CODE**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_Department IN VARCHAR2,

p\_BonusPercentage IN NUMBER

) AS

BEGIN

FOR emp IN (

SELECT EmployeeID, Name, Salary FROM Employees WHERE Department = p\_Department

) LOOP

UPDATE Employees

SET Salary = Salary + (emp.Salary \* p\_BonusPercentage / 100)

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE(

'Bonus applied to ' || emp.Name ||

' | New Salary: ' || (emp.Salary + emp.Salary \* p\_BonusPercentage / 100)

);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Bonus updated for department: ' || p\_Department);

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

SELECT \* FROM Employees;

**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.

**CODE**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) AS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

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

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_FromAccountID;

-- Credit to target

UPDATE Accounts

SET Balance = Balance + p\_Amount,

LastModified = SYSDATE

WHERE AccountID = p\_ToAccountID;

DBMS\_OUTPUT.PUT\_LINE(

'Transferred $' || p\_Amount || ' from Account ' || p\_FromAccountID ||

' to Account ' || p\_ToAccountID

);

END;

/

BEGIN

TransferFunds(1, 2, 200);

END;

/

SELECT \* FROM Accounts;

**OUTPUT**

