**PLSQL\_Exercises**

**Exercise 1: Control Structures:**

Step 1 - First we will create two table “Customers” and “Loans” in the database -

* **Table Customers:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

InterestRate NUMBER(5,2),

Balance NUMBER(10,2),

IsVIP VARCHAR2(5)

);

* **Table Loans:**

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

DueDate DATE

);

**Step 2** – Now we will add sample values to our tables for testing the scenarios.

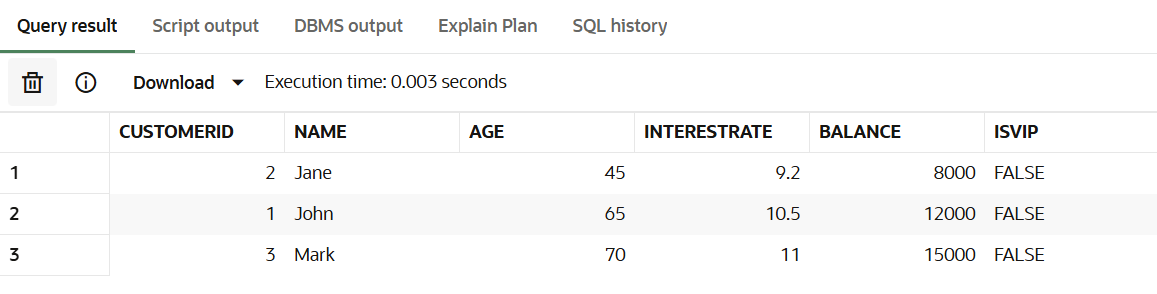
* **Table Customers:**

INSERT INTO Customers VALUES (1, 'John', 65, 10.5, 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Jane', 45, 9.2, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Mark', 70, 11.0, 15000, 'FALSE');

**select \* from customers;**



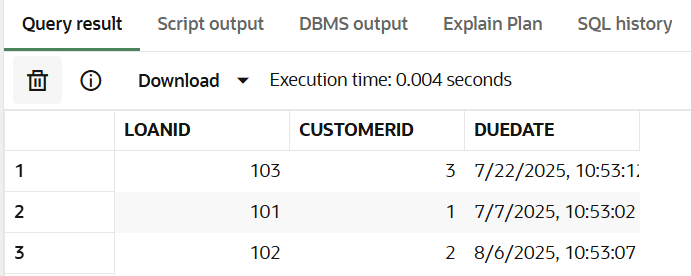
* **Table Loans:**

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

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

INSERT INTO Loans VALUES (103, 3, SYSDATE + 25);

**select \* from loans;**



**Step 3** – Given Scenarios.

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

BEGIN

FOR cust\_rec IN (SELECT CustomerID, InterestRate

FROM Customers

WHERE Age > 60)

LOOP

UPDATE Customers

SET InterestRate = InterestRate - 1

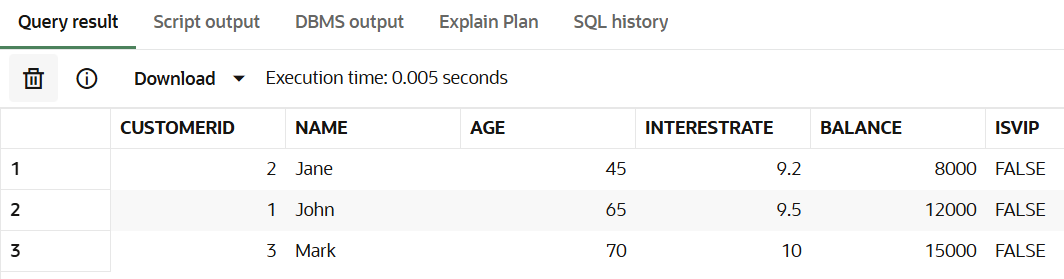
WHERE CustomerID = cust\_rec.CustomerID;

END LOOP;

COMMIT;

END;

**Output:** Select \* from Customers;



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

BEGIN

FOR cust\_rec IN (SELECT CustomerID

FROM Customers

WHERE Balance > 10000)

LOOP

UPDATE Customers

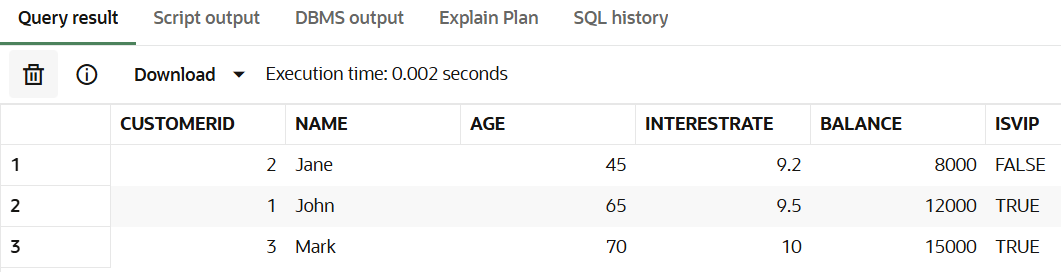
SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_rec.CustomerID;

END LOOP;

COMMIT;

END;

**Output:** Select \* from Customers;

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

BEGIN

FOR loan\_rec IN (

SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

)

LOOP

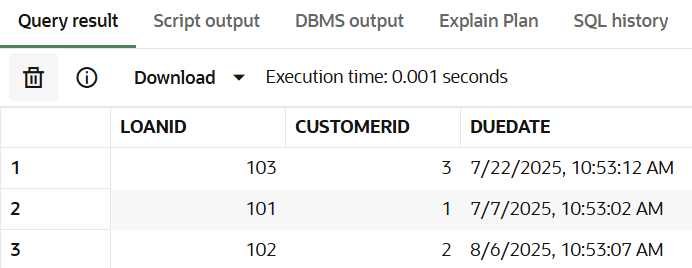
DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.LoanID ||

' for Customer ID ' || loan\_rec.CustomerID ||

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

END LOOP;

END;

**Output:** Select \* from Loans; 

**Exercise 3: Stored Procedures:**

**Step 1** – First we will create the tables – Savings Account, Employees, General Accounts.

* **Table SavingsAccounts**

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(10,2)

);

* **Table Employees**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Department VARCHAR2(50),

Salary NUMBER(10,2)

);

* **Table Accounts (for General Accounts)**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(10,2)

);

**Step 2** – Adding values to the tables for testing the scenarios.

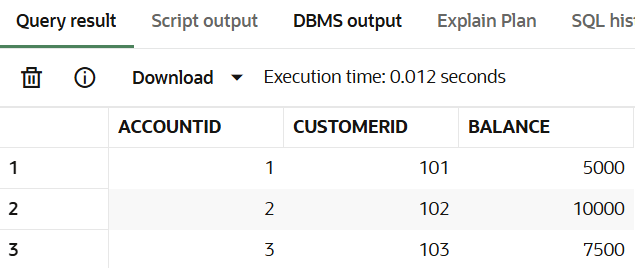
* **Table SavingsAccounts**

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance) VALUES (1, 101, 5000);

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance) VALUES (2, 102, 10000);

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance) VALUES (3, 103, 7500);

**Output:** Select \* from SavingsAccounts;



* **Table Employees**

INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (1, 'Alice', 'Sales', 50000);

INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (2, 'Bob', 'Sales', 55000);

INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (3, 'Charlie', 'IT', 60000);

INSERT INTO Employees (EmployeeID, Name, Department, Salary) VALUES (4, 'Diana', 'IT', 65000);

**Output:** Select \* from Employees;

A screenshot of a computer

AI-generated content may be incorrect.

* **Table Accounts(for General Accounts)**

INSERT INTO Accounts (AccountID, CustomerID, Balance) VALUES (1, 201, 8000);

INSERT INTO Accounts (AccountID, CustomerID, Balance) VALUES (2, 202, 3000);

INSERT INTO Accounts (AccountID, CustomerID, Balance) VALUES (3, 203, 12000);

Output: Select \* from Accounts;

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3** – Given Scenarios.

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

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

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

WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

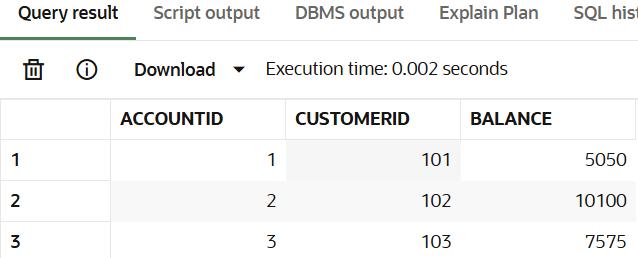
END;

To Execute : BEGIN

ProcessMonthlyInterest;

END;

**Output:** Select \* from SavingsAccounts;



* **Scenario 2 –** The bank wants to implement a bonus scheme for employees based on their performance.

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;

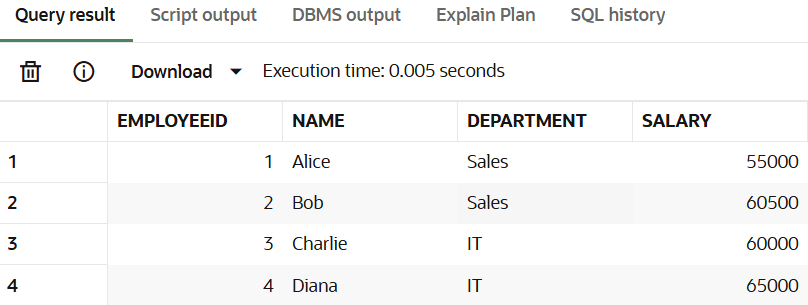
END;

To Execute : BEGIN

UpdateEmployeeBonus('Sales', 10);

END;

**Output:** Select \* from Employees;

****

* **Scenario 3 –** Customers should be able to transfer funds between their accounts.

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 AccountID = p\_from\_account;

IF v\_balance >= p\_amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

ELSE

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

END IF;

END;

To execute: BEGIN

TransferFunds(1, 2, 100);

END;

**Output:** Select \* from Accounts;

