**PL/SQL programming**

**Exercise 1: Control Structures**

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

**o 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.**

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

**o 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.**

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

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

**Table Creation:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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

VALUES (1, 'John Doe', TO\_DATE('1960-05-15', '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'), 1500, SYSDATE);

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

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

**Scenario1:**

declare

cursor senior\_customers is

select c.CustomerID

from Customers c

where MONTHS\_BETWEEN(SYSDATE, c.DOB)/12>60;

begin

for cust in senior\_customers loop

update Loans

set InterestRate=InterestRate-1

where CustomerID=cust.CustomerID;

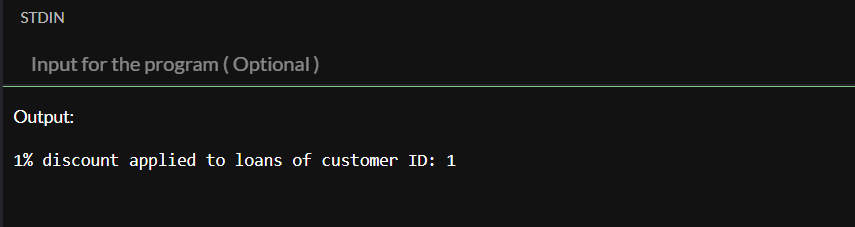
DBMS\_OUTPUT.PUT\_LINE('1% discount applied to loans of customer ID: '||cust.CustomerID);

end loop;

end;

/

**Output:**



**Scenario2:**

**Table Creation:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

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'), 1500, SYSDATE);

-- Add IsVIP column to Customers table

alter table Customers add(IsVIP VARCHAR2(5));

-- Make John's balance 15,000 (VIP)

update Customers set Balance=20000 where CustomerID=1;

-- Make Jane's balance 8000 (Not VIP)

update Customers set Balance=5000 where CustomerID=2;

commit;

**Query:**

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('Customer '||cust.Name||' promoted to VIP.');

else

update Customers

set IsVIP='FALSE'

where CustomerID=cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer '||cust.Name||' is not VIP.');

end if;

end loop;

end;

/

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario3:**

**Table creation:**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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'), 1500, SYSDATE);

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

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

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

VALUES (2, 2, 3000, 6.5, SYSDATE, SYSDATE + 10);

COMMIT;

**Query:**

BEGIN

for rec in(

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

from Loans l

join Customers c on l.CustomerID=c.CustomerID

where l.EndDate between SYSDATE and SYSDATE+30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || rec.Name ||

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

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

'. Please ensure timely payment.');

end LOOP;

end;

/

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**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.**

**Scenario1:**

Table Creation:

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)

);

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'), 1500, 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);

**Query:**

begin

-- Loop through all savings accounts

for acc in(

select AccountID,Balance

from Accounts

where UPPER(AccountType)='SAVINGS'

) LOOP

-- Calculate new balance with 1% interest

update Accounts

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

LastModified=SYSDATE

where AccountID=acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acc.AccountID ||

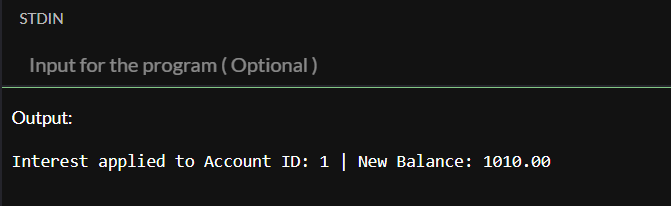
' | New Balance: ' || TO\_CHAR(acc.Balance + (acc.Balance \* 0.01), 'FM9999990.00'));

end LOOP;

end;

/

**Output:**



**Scenario2:**

**Table Creation:**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

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'));

create or replace procedure updateemployeebonus(

p\_department in varchar2,

p\_bonuspercent in number

) is

begin

for emp in (

select employeeid, name, salary

from employees

where upper(department) = upper(p\_department)

) loop

update employees

set salary = salary + (emp.salary \* (p\_bonuspercent / 100))

where employeeid = emp.employeeid;

dbms\_output.put\_line(

'Bonus applied to ' || emp.name ||

' | New Salary: ' || to\_char(emp.salary \* (1 + p\_bonuspercent / 100), 'fm9999990.00')

);

end loop;

end;

/

begin

updateemployeebonus('HR', 10);

end;

/

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario3:**

**Table Creation:**

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)

);

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'), 1500, 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);

create or replace procedure transferfunds(

p\_from\_account in number,

p\_to\_account in number,

p\_amount in number

) is

v\_from\_balance accounts.balance%type;

**Query:**

begin

-- Fetch balance of the source account

select balance into v\_from\_balance

from accounts

where accountid=p\_from\_account;

-- Check if source has enough funds

if v\_from\_balance<p\_amount then

raise\_application\_error(-20001,'Insufficient balance in the source account.');

end if;

-- Deduct from source account

update accounts

set balance=balance-p\_amount,

lastmodified=sysdate

where accountid=p\_from\_account;

-- Credit to destination account

update accounts

set balance=balance+p\_amount,

lastmodified=sysdate

where accountid=p\_to\_account;

dbms\_output.put\_line(

'Transferred ' || p\_amount || ' from Account ID ' || p\_from\_account ||

' to Account ID ' || p\_to\_account

);

end;

/

begin

transferfunds(1, 2, 500);

end;

/

**Output:**

