**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.

**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.

**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:**

create table Customers(

CustomerID number primary key,

Name varchar2(100),

Age number,

Balance number(12,2),

LoanInterestRate number(5,2),

LoanDueDate date,

IsVIP varchar2(5) default 'FALSE'

);

insert into Customers (CustomerID,Name,Age,Balance,LoanInterestRate,LoanDueDate) values (1,'Rama',65,15000,7.5,sysdate+10);

insert into Customers (CustomerID,Name,Age,Balance,LoanInterestRate,LoanDueDate) values (2,'Leela',45,8500,6.0,sysdate+35);

insert into Customers (CustomerID,Name,Age,Balance,LoanInterestRate,LoanDueDate) values (3,'Shiva',70,11000,8.0,sysdate+20);

insert into Customers (CustomerID,Name,Age,Balance,LoanInterestRate,LoanDueDate) values (4,'Radha',68,9000,6.5,sysdate+15);

insert into Customers (CustomerID,Name,Age,Balance,LoanInterestRate,LoanDueDate) values (5,'Lakshmi',50,18000,5.5,sysdate+40);

declare

New\_LoanInterestRate number(5,2);

begin

dbms\_output.put\_line('SCENARIO 1:');

for cust in(

select CustomerID,Name,Age,LoanInterestRate from Customers where Age>60

)loop

update Customers

set LoanInterestRate=LoanInterestRate\*0.99

where CustomerID=cust.CustomerID returning LoanInterestRate into New\_LoanInterestRate;

dbms\_output.put\_line('Discount applied to: Name= '||cust.Name||', Age= '||cust.Age||', New Rate= '||New\_LoanInterestRate);

end loop;

commit;

dbms\_output.put\_line('Discount applied to customers with age above 60');

exception

when others then

rollback;

dbms\_output.put\_line('Error in applying discount to customers with age above 60');

end;

/

declare

New\_IsVIP varchar2(5);

begin

dbms\_output.put\_line('SCENARIO 2:');

for cust in(

select CustomerID,Name,Balance,IsVIP from Customers where Balance>10000

)loop

update Customers

set IsVIP='TRUE'

where CustomerID=cust.CustomerID returning IsVIP into New\_IsVIP;

dbms\_output.put\_line('Marked as VIP: Name= '||cust.Name||', Balance= '||cust.Balance||', VIP status= '||New\_IsVIP);

end loop;

commit;

dbms\_output.put\_line('VIP status is updated for eligible customers');

exception

when others then

rollback;

dbms\_output.put\_line('Error in updating the VIP status to eligible customers');

end;

/

begin

dbms\_output.put\_line('SCENARIO 3:');

for cust in(

select Name,LoanDueDate,LoanInterestRate from Customers where LoanDueDate between sysdate and sysdate+30

)loop

dbms\_output.put\_line('Remainder: Mr/Ms. '||cust.Name||', your loan is due on '||to\_char(cust.LoanDueDate,'DD-MON-YYYY')||

', current interest rate: '|| cust.LoanInterestRate);

end loop;

exception

when others then

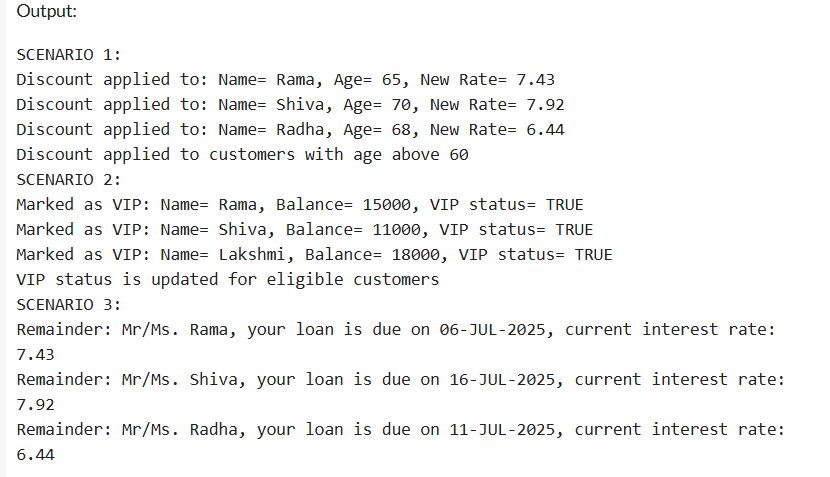
rollback;

dbms\_output.put\_line('Error in sending the Remainder messages to customers who have loan due in next 30 days');

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.

**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.

**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 table Accounts(

AccountID number primary key,

CustomerID number,

AccountType varchar2(20),

Balance number(12,2)

);

create table Employees(

EmployeeID number primary key,

Name varchar2(100),

Department varchar2(50),

Salary number(10,2)

);

insert into Accounts(AccountID,CustomerID,AccountType,Balance) values (101,1,'Savings',10000.00);

insert into Accounts(AccountID,CustomerID,AccountType,Balance) values (102,2,'Savings',20000.00);

insert into Accounts(AccountID,CustomerID,AccountType,Balance) values (103,3,'Current',9000.00);

insert into Accounts(AccountID,CustomerID,AccountType,Balance) values (104,1,'Current',5000.00);

insert into Accounts(AccountID,CustomerID,AccountType,Balance) values (105,4,'Savings',18000.00);

insert into Employees(EmployeeID,Name,Department,Salary) values (301,'Rama','IT',60000.00);

insert into Employees(EmployeeID,Name,Department,Salary) values (302,'Leela','HR',50000.00);

insert into Employees(EmployeeID,Name,Department,Salary) values (303,'Radha','IT',70000.00);

insert into Employees(EmployeeID,Name,Department,Salary) values (304,'Shiva','HR',45000.00);

insert into Employees(EmployeeID,Name,Department,Salary) values (305,'Lakshmi','Finance',80000.00);

create or replace procedure ProcessMonthlyInterest as

begin

update Accounts

set Balance=Balance+(Balance\*0.01)

where AccountType='Savings';

commit;

dbms\_output.put\_line('SCENARIO 1:');

dbms\_output.put\_line('updated Savings Accounts details:');

for acc in(

select AccountID,CustomerID,Balance from Accounts where AccountType='Savings'

)loop

dbms\_output.put\_line('AccountID: '||acc.AccountID||', CustomerID: '||acc.CustomerID||', New Balance: '||acc.Balance);

end loop;

exception

when others then

rollback;

dbms\_output.put\_line('Error applying monthly interest to the saving accounts');

end;

/

begin

ProcessMonthlyInterest();

end;

/

create or replace procedure UpdateEmployeeBonus(

dept in varchar2,

bonus\_percent in number)

as

begin

update Employees

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

where Department=dept;

commit;

dbms\_output.put\_line('SCENARIO 2');

dbms\_output.put\_line('Bonus applied to '||dept||' department. Updated employee details:');

for emp in(

select EmployeeID,Name,Salary from Employees where department=dept

)loop

dbms\_output.put\_line('EmployeeID: '||emp.EmployeeID||', Name:'||emp.Name||', New Salary: '||emp.Salary);

end loop;

exception

when others then

rollback;

dbms\_output.put\_line('Error updating employee bonus');

end;

/

begin

UpdateEmployeeBonus('IT',10);

dbms\_output.put\_line('SCENARIO 3');

end;

/

create or replace procedure TransferFunds(

from\_account in number,

to\_account in number,

amount in number

)as

from\_balance number;

to\_balance number;

begin

select Balance into from\_balance from Accounts where AccountID=from\_account

for update;

select Balance into to\_balance from Accounts where AccountID=to\_account

for update;

if from\_balance<amount then

dbms\_output.put\_line('Error: Insufficient balance in source account');

return;

end if;

update Accounts

set Balance=Balance-amount

where AccountID=from\_account;

update Accounts

set Balance=Balance+amount

where AccountID=to\_account;

commit;

dbms\_output.put\_line('Transfer successfull');

for acc in(

select AccountID,CustomerID,Balance from Accounts where AccountID in(from\_account,to\_account)

)loop

dbms\_output.put\_line('AccountID: '||acc.AccountID||', CustomerID: '||acc.CustomerID||', Balance: '||acc.Balance);

end loop;

exception

when NO\_DATA\_FOUND then

rollback;

dbms\_output.put\_line('One of the Account IDs does not exist');

when others then

rollback;

dbms\_output.put\_line('Error during Transfer:');

end;

/

begin

TransferFunds(101,103,3000);

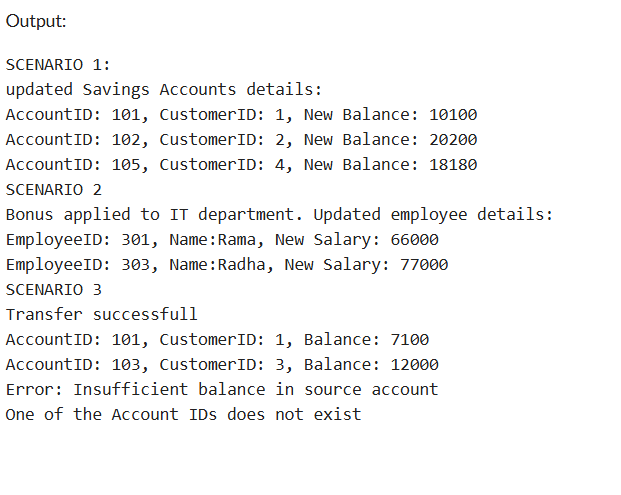
TransferFunds(102,104,25000);

TransferFunds(103,107,2000);

end;

/

**Output:**

****