## SCHEMA:

## Customers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CustomerID | Name | Date of Birth | Balance | Last Modified |
| 1 | Arjun Kumar | 15-MAR-1985 | 15,000 | 26-JUN-2025 |
| 2 | Priya Sharma | 10-AUG-1990 | 25,000 | 26-JUN-2025 |
| 3 | Rahul Verma | 01-DEC-1975 | 30,000 | 26-JUN-2025 |
| 4 | Sundaram Iyer | 20-APR-1955 | 50,000 | 26-JUN-2025 |
| 5 | Lakshmi Reddy | 30-SEP-1959 | 60,000 | 26-JUN-2025 |

## Accounts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AccountID | CustomerID | AccountType | Balance | Last Modified |
| 1 | 1 | Savings | 1000 | 26-JUN-2025 |
| 2 | 2 | Checking | 1500 | 26-JUN-2025 |
| 3 | 3 | Savings | 2500 | 26-JUN-2025 |
| 4 | 4 | Current | 5000 | 26-JUN-2025 |
| 5 | 5 | Fixed Deposit | 100000 | 26-JUN-2025 |

## Loans

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LoanID | CustomerID | LoanAmount | InterestRate | StartDate | EndDate |
| 1 | 1 | 5000 | 5 | 26-JUN-2025 | 26-JUN-2030 |
| 2 | 2 | 50000 | 7.5 | 01-JAN-2025 | 01-JAN-2027 |
| 3 | 3 | 100000 | 8 | 15-MAR-2025 | 02-JUL-2025 |
| 4 | 4 | 25000 | 6.5 | 10-JUN-2025 | 10-JUN-2026 |
| 5 | 5 | 80000 | 7 | 20-JUL-2025 | 20-JUL-2029 |

## Transactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TransactionID | AccountID | TransactionDate | Amount | TransactionType |
| 1 | 1 | 26-JUN-2025 | 200 | Deposit |
| 2 | 2 | 26-JUN-2025 | 300 | Withdrawal |
| 3 | 3 | 26-JUN-2025 | 1000 | Credit |
| 4 | 4 | 26-JUN-2025 | 500 | Debit |
| 5 | 5 | 26-JUN-2025 | 2000 | Credit |

## Employees

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EmployeeID | Name | Position | Salary | Department | HireDate |
| 1 | Alice Johnson | Manager | 70000 | HR | 15-JUN-2015 |
| 2 | Bob Brown | Developer | 60000 | IT | 20-MAR-2017 |

**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.

**PL/SQL BLOCK:**

DECLARE

v\_cid customers.customerid%type;

v\_name customers.name%type;

v\_dob customers.dob%type;

v\_cid1 loans.customerid%type;

v\_interest loans.interestrate%type;

v\_age integer;

CURSOR c\_cust IS

SELECT c.customerid AS cust\_id,

c.name AS cust\_name,

c.dob AS cust\_dob,

l.customerid AS loan\_cust\_id,

l.interestrate AS loan\_interest

FROM customers c

JOIN loans l ON l.customerid = c.customerid;

BEGIN

OPEN c\_cust;

LOOP

FETCH c\_cust INTO v\_cid, v\_name, v\_dob, v\_cid1, v\_interest;

EXIT WHEN c\_cust%NOTFOUND;

v\_age := FLOOR((SYSDATE - v\_dob) / 365.25);

IF v\_age > 60 THEN

v\_interest := v\_interest - 1.0;

UPDATE loans

SET interestrate = v\_interest

WHERE customerid = v\_cid1;

DBMS\_OUTPUT.PUT\_LINE('Customer name: ' || v\_name ||' New Interest rate: ' || v\_interest);

END IF;

END LOOP;

CLOSE c\_cust;

COMMIT;

EXCEPTION

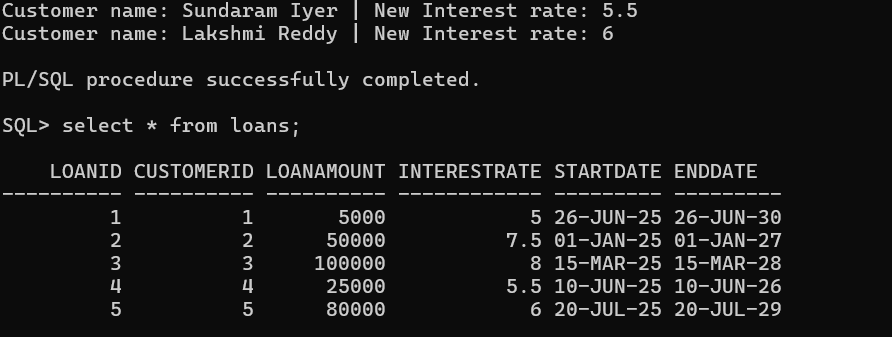
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some other error occurred: ' || SQLERRM);

END;

/

**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.

**PL/SQL BLOCK:**

DECLARE

isVip varchar2(10):='false';

BEGIN

FOR cust IN (SELECT customerid,name,balance FROM customers) LOOP

if(cust.balance>10000) THEN

isVip:='true';

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER ID: '||cust.customerid||' | CUSTOMER NAME: '||cust.name||

' | BALANCE: '||cust.balance||' | IsVip: '||isvip);

END IF;

END LOOP;

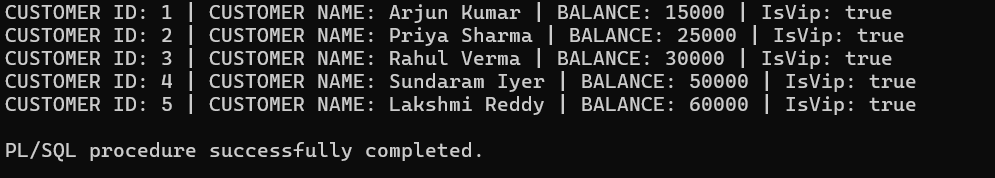
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some other error occurred: ' || SQLERRM);

END;

**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.

**PL/SQL BLOCK:**

BEGIN

FOR rec IN (

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer "' || rec.Name || '" (ID: ' || rec.CustomerID ||

') has a loan (Loan ID: ' || rec.LoanID || ') due on ' ||

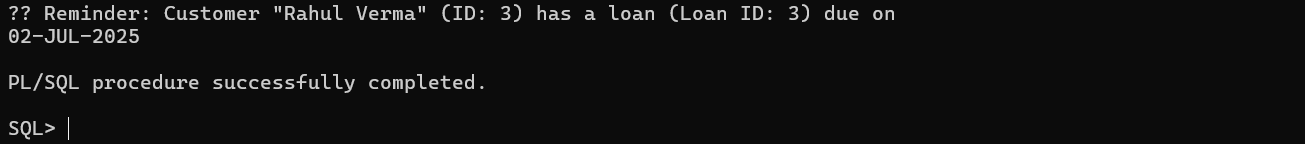
TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

END LOOP;

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.

**PROCEDURE:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

acid accounts.accountid%type;

newBalance accounts.balance%type;

interest number;

BEGIN

FOR rec IN (SELECT accountid,accounttype,balance from accounts WHERE accounttype='Savings') LOOP

interest:=rec.balance\*(1/100);

newBalance:=rec.balance+interest;

UPDATE accounts

SET balance=newBalance WHERE accountid=rec.accountid;

END LOOP;

EXCEPTION

WHEN others THEN

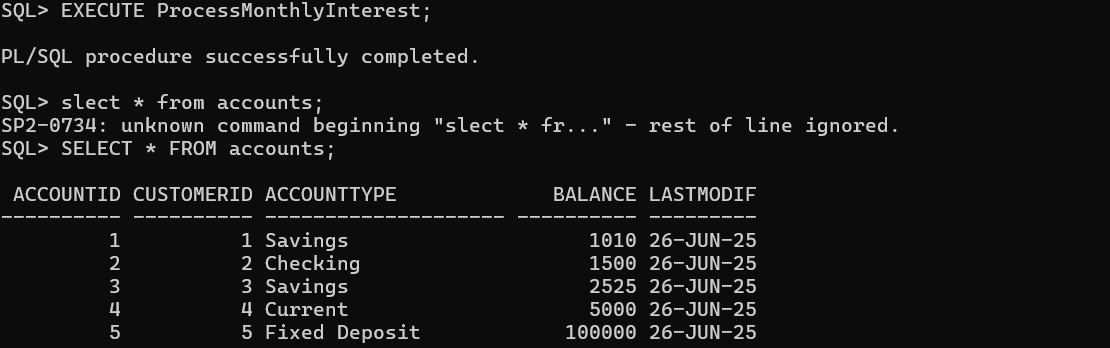
DBMS\_OUTPUT.PUT\_LINE('some error occurred..');

END;

/

EXECUTE ProcessMonthlyInterest;

**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.

**PROCEDURE:**

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;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to all employees in department: ' || p\_department);

EXCEPTION

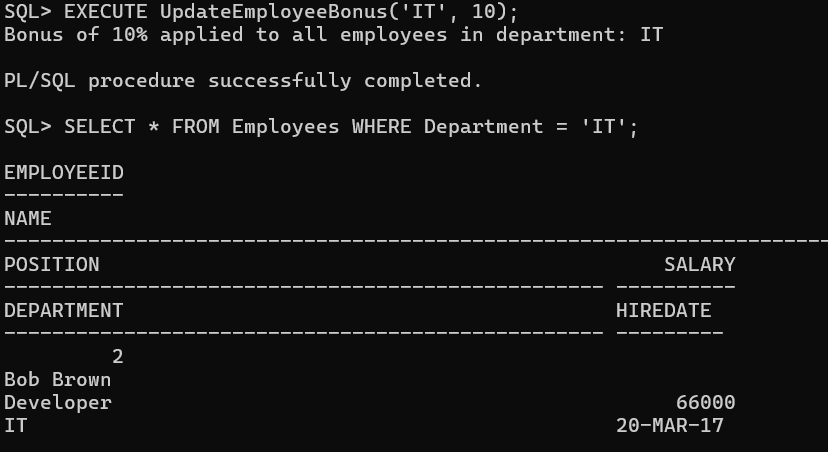
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

END;

/

**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.

**PROCEDURE:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_from\_balance NUMBER;

BEGIN

IF p\_amount <= 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer amount must be greater than 0.');

RETURN;

END IF;

SELECT Balance INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_account\_id;

IF v\_from\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in source account.');

RETURN;

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account\_id;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account\_id;

DBMS\_OUTPUT.PUT\_LINE( p\_amount || ' transferred from Account ID ' ||

p\_from\_account\_id || ' to Account ID ' || p\_to\_account\_id);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both account IDs not found.');

WHEN OTHERS THEN

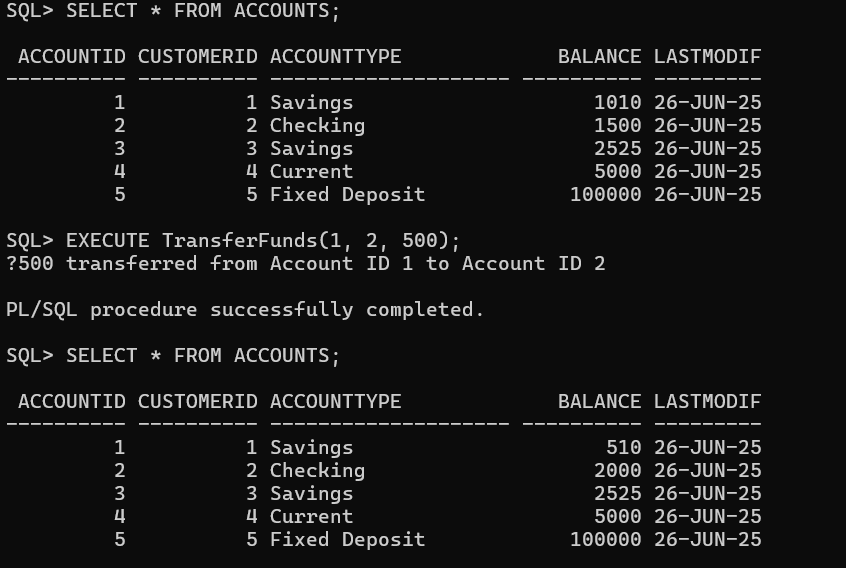
DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

EXECUTE TransferFunds(1, 2, 500);

**OUTPUT:**

****