Banking Management System

Designed and implemented relational database tables

create table customers (customer_id number(3) primary key, name varchar2(10), city varchar2(15), contact number);

create table accounts (account_id   number(4) primary key, customer_id number(3) references customerss (customer_id), account_type varchar2(15), balance number);

create table transactions (txn_id number(4) primary key, account_id number(4) references accounts(account_id), txn_type varchar2(15), amount number, txn_date date);

Create Sequence

create sequence seq_customer start with 1 increment by 1;

create sequence seq_account start with 1001 increment by 1;

create sequence seq_txn start with 1 increment by 1;

Procedure

CREATE CUSTOMER

create or replace procedure create_customer (

```
   p_name    in varchar2,
   p_city    in varchar2,
   p_contact in number
)
AS
begin
   insert into customerss
   values (seq_customer.NEXTVAL, p_name, p_city, p_contact);

    dbms_output.put_line('Customer ' || p_name || ' inserted');
    dbms_output.put_line('Customer ' || p_city || ' inserted');
    dbms_output.put_line('Customer ' || p_contact || ' inserted');

end;
/
```

EXECUTE THIS PROCESS

```
begin
  create_customer('Rahul', 'Delhi', 9876543210);
  create_customer('Amit', 'Mumbai', 9876543211);
  create_customer('Priya', 'Bangalore', 9876543212);
end;
/
```

OPEN_ACCOUNT

```
create or replace procedure open_account (
   p_customer_id  in varchar2,
   p_account_type in varchar2,
   p_balance      in number
)
AS
begin
   insert into accounts
   values(seq_account.NEXTVAL, p_customer_id, p_account_type,
p_balance);

  dbms_output.put_line ('Account opened successfully');
   dbms_output.put_line ('Customer ID   : ' || p_customer_id);
   dbms_output.put_line ('Account Type  : ' || p_account_type);
   dbms_output.put_line ('Opening Bal.  : ' || p_balance);
END;
/
```

EXECUTE THIS PROCEDURE

```
begin
  open_account(1, 'Savings', 5000);
  open_account(2, 'Current', 10000);
  open_account(3, 'Savings', 8000);
```

```
end;
/
```

ADD_TRANSACTION

```
create or replace procedure add_transaction (
   p_account_id in number,
   p_txn_type   in varchar2,
   p_amount     in number
)
as
begin
   insert into transactions (
      txn_id,
      account_id,
      txn_type,
      amount,
      txn_date
   )
   values (
      seq_txn.nextval,
      p_account_id,
      upper(p_txn_type),
      p_amount,
      sysdate
   );
end;
/
```

CREATE PROCEDURE DEPOSITE MONEY

```
create or replace procedure deposit_money (
   p_account_id in number,
   p_amount     in number
)
is
begin
   -- update balance
   update accounts
   set balance = balance + p_amount
   where account_id = p_account_id;

   -- check if account exists
   if sql%rowcount = 0 then
      raise_application_error(-20001, 'account not found');
   end if;

   -- log transaction
   add_transaction(p_account_id, 'deposit', p_amount);
   dbms_output.put_line('deposit successful');

exception
   when others then
      dbms_output.put_line('error: ' || sqlerrm);
end;
/
```

CREATE PROCEDURE WITHDRAW MONEY

```
create or replace procedure withdraw_money (
   p_account_id in number,
   p_amount     in number
)
as
   v_balance number;
begin
   -- get current balance
   select balance
   into   v_balance
   from   accounts
   where  account_id = p_account_id;

   -- check for sufficient balance
   if v_balance < p_amount then
```

```
        raise_application_error(-20002, 'insufficient balance');
    end if;

    -- deduct balance
    update accounts
    set    balance = balance - p_amount
    where  account_id = p_account_id;

    -- log transaction
    add_transaction(p_account_id, 'withdraw', p_amount);

    dbms_output.put_line('withdrawal successful');

exception
    when no_data_found then
        dbms_output.put_line('invalid account number');
    when others then
        dbms_output.put_line('error: ' || sqlerrm);
end;
/
```

TRIGGER

```
create or replace trigger check_balance
before update of balance on accounts
```

```
for each row
begin
   if :NEW.balance < 0 THEN
      RAISE_APPLICATION_ERROR(
         -20001,
         'Insufficient balance. Withdrawal not allowed.'
      );
   end if;
end;
/
```

function

```
create or replace function get_balance (
   p_account_id IN number
)
return number
IS
   v_balance accounts.balance%TYPE;
begin
   select  balance into v_balance from accounts where account_id =
p_account_id;
   return v_balance;
end;
/
```

How to be work

UPDATE AMOUNT DEPOSITE

```
begin
       deposit_money(1001, 5000);
     end;
     /
```

UPDATE AMOUNT WITHDRAW

```
begin
   withdraw_money(1001,1000);
  end;
  /
```

CHECK UPDATED AMOUNT BALANCE

```
Select * from accounts where  account_id = 1001;
```

CHECK TRANSACTIONS HISTORY

```
 Select * from transactions;
```

CHECK TRIGGER NEGATIVE BALANCE UPDATE

```
Update  accounts
SET balance = balance - 999999
WHERE account_id = 1001;
```

CHECK CURRENT BALANCE USING FUNCTION

```
Select  get_balance(1001) AS balance
From  dual;
```