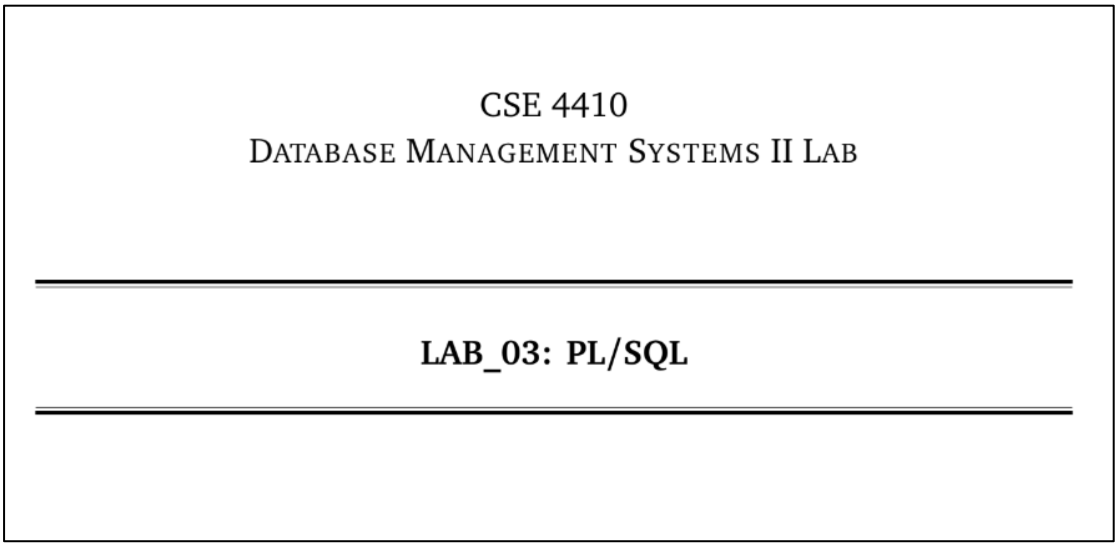
LAB REPORT



CSE 4410

DATABASE MANAGEMENT SYSTEMS II LAB

LAB_03: PL/SQL

NAME: CHOWDHURY ASHFAQ

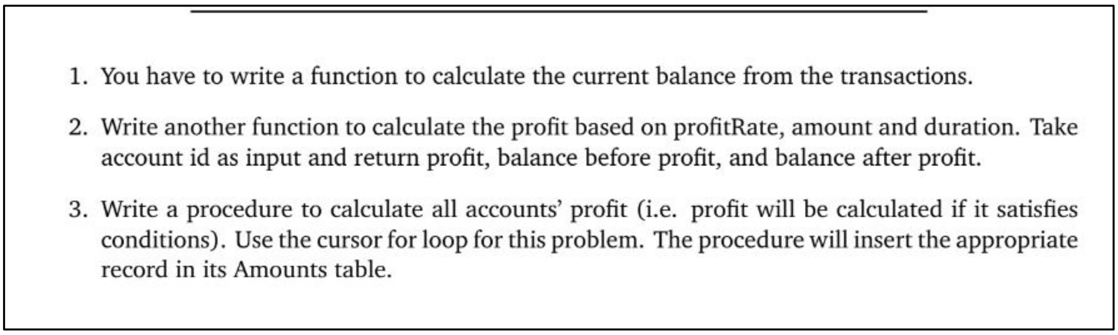
STUDENT ID: 200042123

PROGRAM: SWE

GROUP: 1A

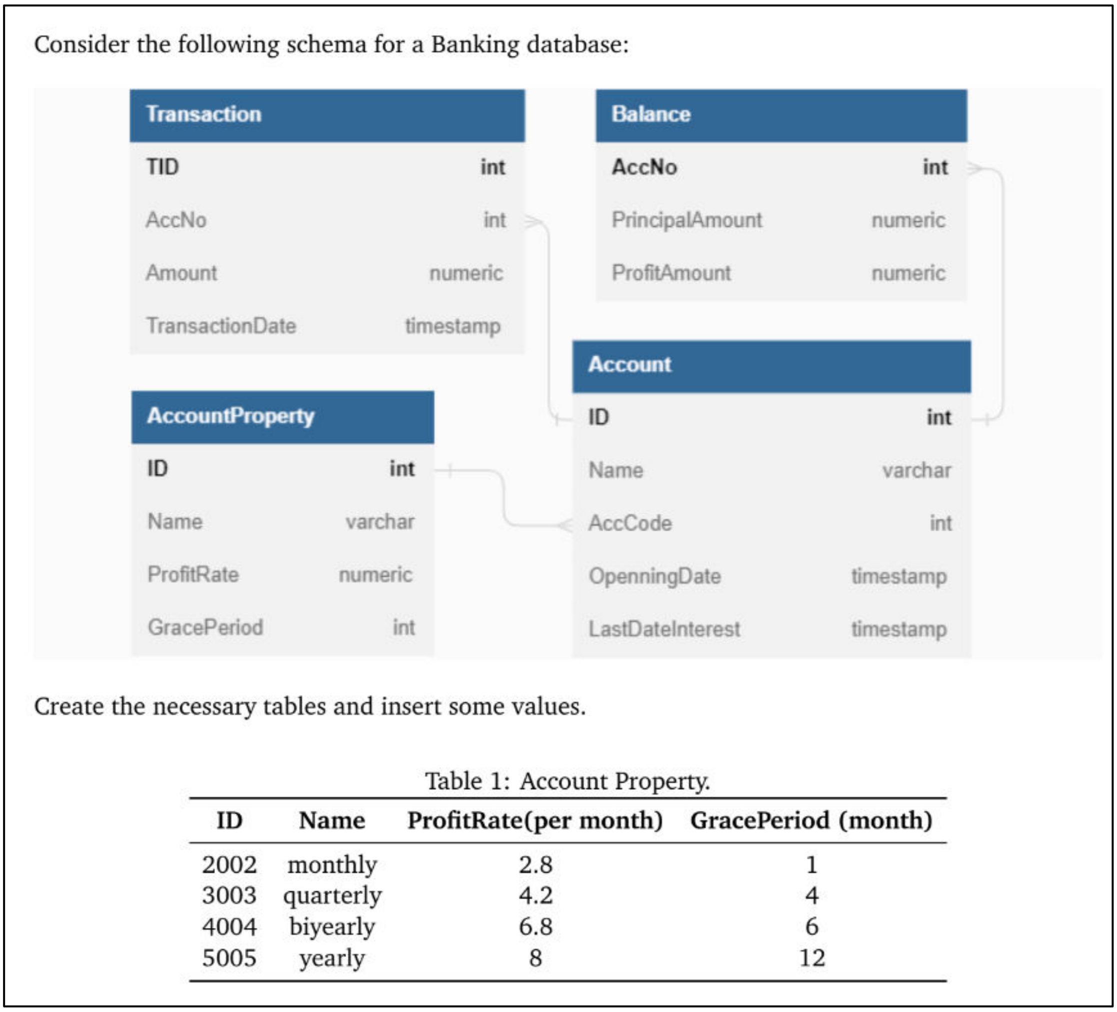
DATE: 03/02/23

Tasks:



1. You have to write a function to calculate the current balance from the transactions.
2. Write another function to calculate the profit based on profitRate, amount and duration. Take account id as input and return profit, balance before profit, and balance after profit.
3. Write a procedure to calculate all accounts' profit (i.e. profit will be calculated if it satisfies conditions). Use the cursor for loop for this problem. The procedure will insert the appropriate record in its Amounts table.

Analysis of the problem:



Consider the following schema for a Banking database:

Transaction	
TID	int
AccNo	int
Amount	numeric
TransactionDate	timestamp

Balance	
AccNo	int
PrincipalAmount	numeric
ProfitAmount	numeric

AccountProperty	
ID	int
Name	varchar
ProfitRate	numeric
GracePeriod	int

Account	
ID	int
Name	varchar
AccCode	int
OpenningDate	timestamp
LastDateInterest	timestamp

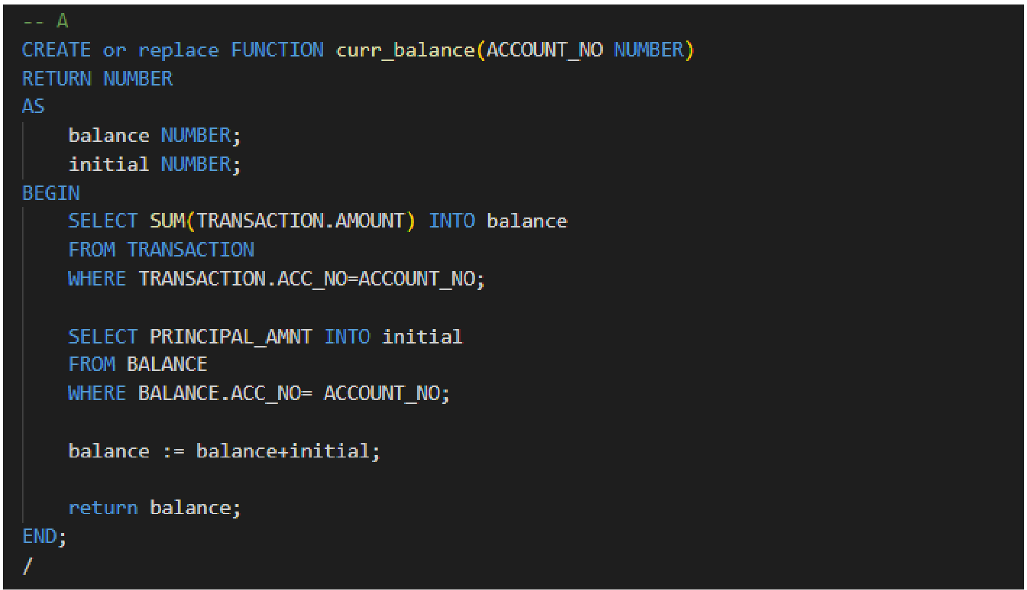
Create the necessary tables and insert some values.

Table 1: Account Property.

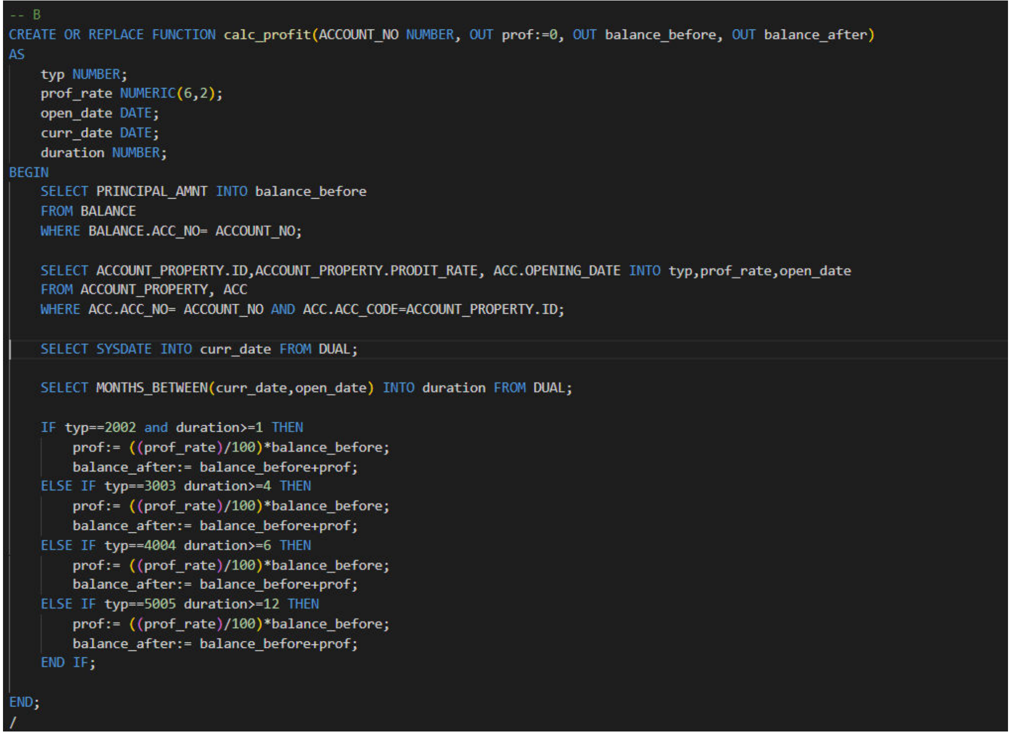
ID	Name	ProfitRate(per month)	GracePeriod (month)
2002	monthly	2.8	1
3003	quarterly	4.2	4
4004	biyearly	6.8	6
5005	yearly	8	12

The above scenario was given which is of a b=Banking database. We had to write 2 functions and 1 procedure. The function of them where pre-defined.

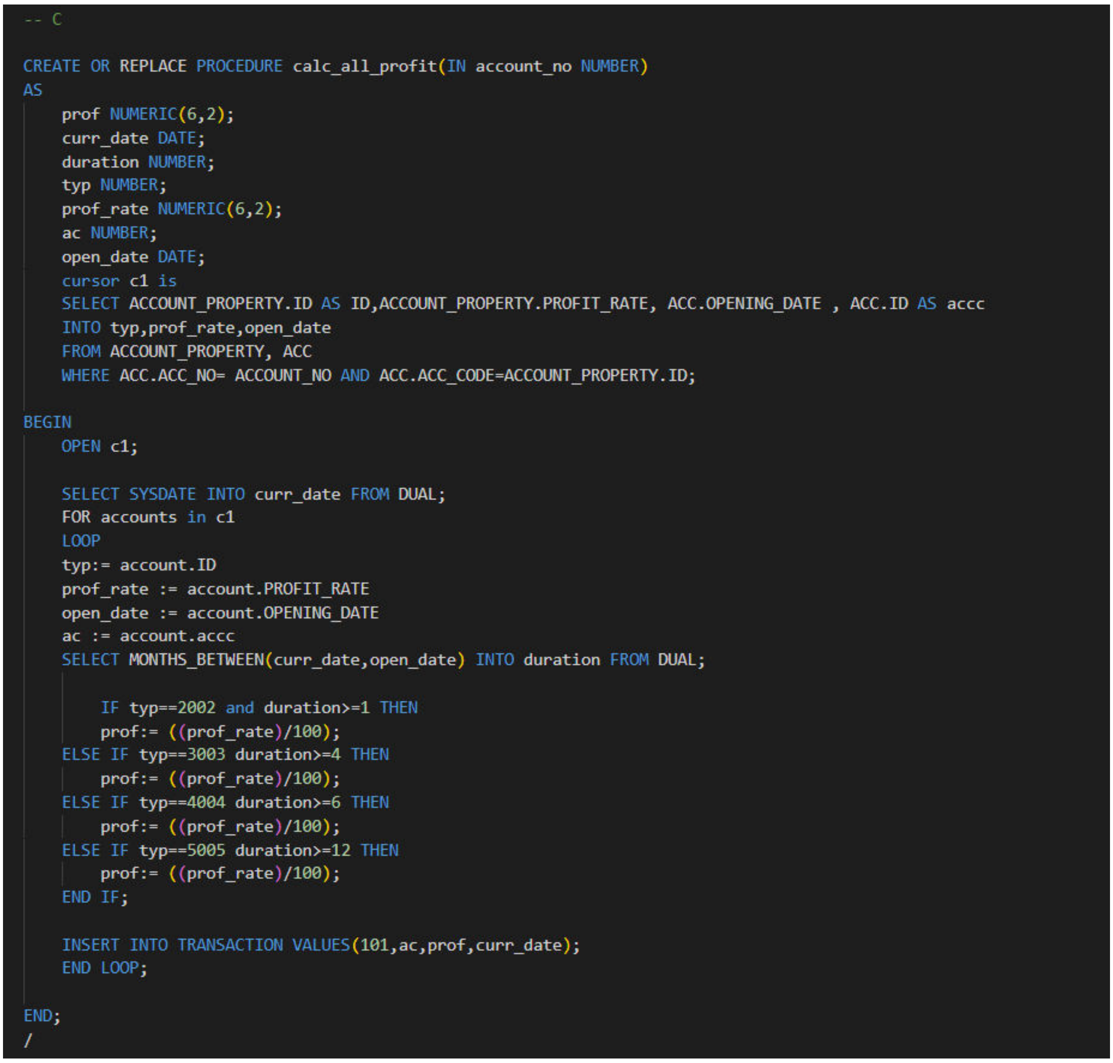
Solution:



```
-- A
CREATE or replace FUNCTION curr_balance(ACCOUNT_NO NUMBER)
RETURN NUMBER
AS
    balance NUMBER;
    initial NUMBER;
BEGIN
    SELECT SUM(TRANSACTION.AMOUNT) INTO balance
    FROM TRANSACTION
    WHERE TRANSACTION.ACC_NO=ACCOUNT_NO;

    SELECT PRINCIPAL_AMNT INTO initial
    FROM BALANCE
    WHERE BALANCE.ACC_NO= ACCOUNT_NO;

    balance := balance+initial;

    return balance;
END;
/
```



```
-- B
CREATE OR REPLACE FUNCTION calc_profit(ACCOUNT_NO NUMBER, OUT prof:=0, OUT balance_before, OUT balance_after)
AS
    typ NUMBER;
    prof_rate NUMERIC(6,2);
    open_date DATE;
    curr_date DATE;
    duration NUMBER;
BEGIN
    SELECT PRINCIPAL_AMNT INTO balance_before
    FROM BALANCE
    WHERE BALANCE.ACC_NO= ACCOUNT_NO;

    SELECT ACCOUNT_PROPERTY.ID,ACCOUNT_PROPERTY.PRODIT_RATE, ACC.OPENING_DATE INTO typ,prof_rate,open_date
    FROM ACCOUNT_PROPERTY, ACC
    WHERE ACC.ACC_NO= ACCOUNT_NO AND ACC.ACC_CODE=ACCOUNT_PROPERTY.ID;

    SELECT SYSDATE INTO curr_date FROM DUAL;

    SELECT MONTHS_BETWEEN(curr_date,open_date) INTO duration FROM DUAL;

    IF typ==2002 and duration>=1 THEN
        prof:= ((prof_rate)/100)*balance_before;
        balance_after:= balance_before+prof;
    ELSE IF typ==3003 duration>=4 THEN
        prof:= ((prof_rate)/100)*balance_before;
        balance_after:= balance_before+prof;
    ELSE IF typ==4004 duration>=6 THEN
        prof:= ((prof_rate)/100)*balance_before;
        balance_after:= balance_before+prof;
    ELSE IF typ==5005 duration>=12 THEN
        prof:= ((prof_rate)/100)*balance_before;
        balance_after:= balance_before+prof;
    END IF;

END;
/
```



```
-- C

CREATE OR REPLACE PROCEDURE calc_all_profit(IN account_no NUMBER)
AS
    prof NUMERIC(6,2);
    curr_date DATE;
    duration NUMBER;
    typ NUMBER;
    prof_rate NUMERIC(6,2);
    ac NUMBER;
    open_date DATE;
    cursor c1 is
    SELECT ACCOUNT_PROPERTY.ID AS ID,ACCOUNT_PROPERTY.PROFIT_RATE, ACC.OPENING_DATE , ACC.ID AS accc
    INTO typ,prof_rate,open_date
    FROM ACCOUNT_PROPERTY, ACC
    WHERE ACC.ACC_NO= ACCOUNT_NO AND ACC.ACC_CODE=ACCOUNT_PROPERTY.ID;

BEGIN
    OPEN c1;

    SELECT SYSDATE INTO curr_date FROM DUAL;
    FOR accounts in c1
    LOOP
    typ:= account.ID
    prof_rate := account.PROFIT_RATE
    open_date := account.OPENING_DATE
    ac := account.accc
    SELECT MONTHS_BETWEEN(curr_date,open_date) INTO duration FROM DUAL;

        IF typ==2002 and duration>=1 THEN
        prof:= ((prof_rate)/100);
    ELSE IF typ==3003 duration>=4 THEN
        prof:= ((prof_rate)/100);
    ELSE IF typ==4004 duration>=6 THEN
        prof:= ((prof_rate)/100);
    ELSE IF typ==5005 duration>=12 THEN
        prof:= ((prof_rate)/100);
    END IF;

    INSERT INTO TRANSACTION VALUES(101,ac,prof,curr_date);
    END LOOP;

END;
/
```

Explanation:

1) This is a PL/SQL function that calculates the current balance of a bank account. The function takes the account number as an input and returns the balance as a number. The function starts by finding the sum of all the transactions made on the account and stores it in the "balance" variable. Next, it retrieves the initial balance of the account from the "BALANCE" table and stores it in the "initial" variable. Finally, it adds the initial balance to the balance of all transactions to get the current balance, which is then returned as the result of the function.

2) This is a PL/SQL function that calculates the profit and balance of a bank account. The function takes the account number as an input and returns the profit, balance before and after the profit calculation as output parameters. The function starts by retrieving the initial balance

of the account from the "BALANCE" table and storing it in the "balance_before" output parameter. Next, it retrieves the account type, profit rate and the opening date of the account from the "ACCOUNT_PROPERTY" and "ACC" tables. It also retrieves the current date from the database. The function then calculates the duration between the current date and the opening date of the account. Based on the account type and duration, the function calculates the profit and updates the balance after the profit calculation in the "balance_after" output parameter. Finally, it returns the profit as the result of the function.

3)
The "calc_all_profit" procedure calculates the profit for all bank accounts with the given account number.

The procedure has a cursor "c1" which selects the account properties and opening date of the account from the ACCOUNT_PROPERTY and ACC tables. The procedure then loops through each account returned by the cursor, calculates the profit based on the type of account and duration, and inserts a new transaction into the TRANSACTION table with the calculated profit and current date.

Problems Faced:
Syntax is always a problem.