Report for Assignment 2

ISYS224: DATABASE SYSTEMS (2018)

Department of Computing Macquarie University

Submitted by: Chun Hei Matthew Lee

Student ID: 45282188

Date Submitted:

Declaration:

In submitting this report, I ,Chun Hei Matthew Lee, declare that this is my own work.

Task 1 CREATING AND POPULATING DATABASE TABLES

no errors

Task 2 PROCEDURE FOR LOAN REPAYMENT

(a)

Procedure Implementation

```
definition //
drop procedure if exists Repay_Loan //
create procedure Repay_Loan(in from_BSB varchar(10), in from_account varchar(10), in to_loan varchar(10), in amount double)
   Declare no_row_found int default 0;#error handling variable for no row found
   Declare saving amount double: #variable to store how much you have in your from account
  #find the corresponding loanBSB and accountNo attached by the loanID Declare loan_BSB varchar(10); Declare loan_Acc varchar(10); Declare loan_left double;
   Declare commonOwner int default 0;#store the number of common owner of both accounts
   declare continue handler for not found set no row found =1:
   select_AccountBal into saving_amount #select the balance from the saving account
   where BSB=from_BSB and AccountNo = from_account;
  #select the corresponding loanAcct for loanID select BSB, AccountNo, AccountBal into loan_BSB, loan_Acc, loan_left from T_Loan, T_Account where (Loan_AccountBSB=BSB and Loan_AcctNo=AccountNo) and to_loan = LoanID;
  signal sqlstate '45000' set message_text='accounts not found'; end if;
  if(no_row_found) then
   #this query finds the number of CustomerID which is the owner of both accounts-loan account and from_acc
   select count(Customer_ID) into commonOwner
   from T_Own
where Account_BSB=from_BSB and
   Account_No=from_account and
(Customer_ID in (select Customer_ID
from T_Own
   where Account_BSB=loan_BSB and Account_No = loan_Acc));
```

```
if(commonOwner=0) then#raise an error if no common Owners found
signal sqlstate '45000' set message_text = 'the owner of account and owner of loan account are not matched';
elseif (amount >saving_amount) then
signal sqlstate '45000' set message_text = 'not enough balance to transfer from account';
else

update T_Account
set AccountBal = AccountBal+amount
where BSB=loan_BSB and AccountNo=loan_Acc;

update T_Account
set AccountBal = saving_amount-amount
where BSB=from_BSB and AccountNo=from_account;

INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
VALUES (to_loan,amount,NOW());

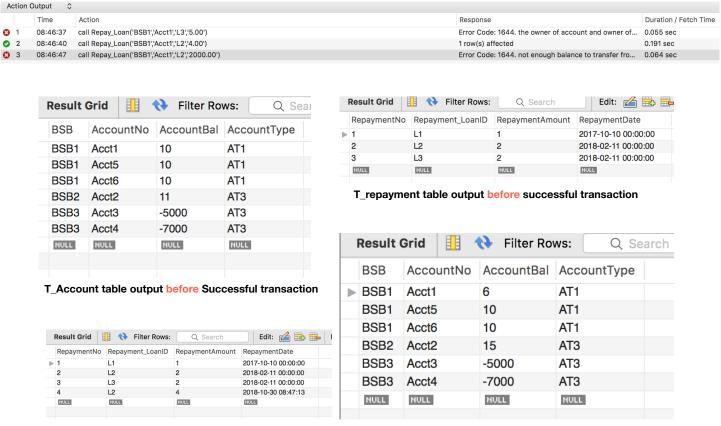
end if;
end //
delimiter:
```

(b)

Procedure Testing

```
call Repay_Loan('BSB1','Acct1','L3','5.00');#customerID are different call Repay_Loan('BSB1','Acct1','L2','4.00');#successful transaction call Repay_Loan('BSB1','Acct1','L2','2000.00');#amount is more than savingBalance
```

Result output



T_Repayment table out after successful transaction

T_Account table output after successful transaction

Assumption for task 2

- 1. Test code has not been ran in the sql script, so you have to check please run the code
- 2. no need to apply an error handler for multiple row retrieval using select...into as the BSB and accountNo as composite are always unique in the table as assumption.

Task 3 DATABASE TRIGGER

- i. ** Rationale behind the type of trigger used ** I would choose before insert trigger for this purpose because it is more easier to validate criteria of all 5 business rules for each new insertion before inserting, and it is more logical to validate data before insert to maintain integrity of the database and consistency of the business rules. I choose to put the trigger in T_Loan because customer is allowed to have many accounts but only one loan account can be attached by maximum one loan. Having a trigger on T_Loan allows me to validate before inserting record based on the 5 business rules.
- ii. **How the trigger is tested ** I will test this trigger based on 5 business rule, for each rule I would try to provide a successful case and a fail case.**Home Loan** I use C1 as my tested target, and he has no homeLoans attached at the beginning. So I insert three home loan records to show it is okay to insert home loan, and when I insert the fourth home loan, I should get a message as C1 has 3 home loans already and fourth one is not allowed. **Individual Loan** is my second check. Since previous three individual home Loans are inserted, I can now can insert the fifth individual loan as C1 was attached to an individual at the start. The sixth one would be failed as shown below (end of this task) because maximum number of individual account is 5. **Personal loan** C1 was attached with a personal loan at the start, the second case I insert would be failed(evidence shown end of this task).

TotalLoanAmount I insert a loan with big loan amount exceeding the maximum of

\$10000000, which is shown as fail in the diagram at the end of this task. **Maximum loan account** up until this point, in the database, there is 6 loan accounts for C1 assuming the previous successful insertion are ran. So inserting 7th and 8th loan account{ not violating other business rules} should be fine and the 9th insertion would be shown as fail in the diagram at the end of this task.

iii. **Code for implementing the trigger**

```
delimiter //
drop trigger if exists task3Trigger //
create trigger task3Trigger
before insert on T_Loan#validate after insertion
for each row
declare v_finished boolean default 0;#to track end of cursor declare raiseError int default 0;
                                            declare temp_CID varchar(10); #used to indicate current customerID in the cursor
                                           declare temp_homeLoan int;#used to indicate number of home loans possesed by a customerID declare personalLoan_count int;#used to indicate number of home loans possesed by a customerID declare temp_totalLoanAmt decimal(11,2);#used to indicate total aamount of all loans possessed by a customerID declare temp_IndividualLoan int; #used to indicate number of individual loans possessed by a customerID declare temp_totalLoan int;#used to indicate number of total loans possessed by a customerID
                                             declare msg varchar(100); #set error message
                                           declare owner_rec cursor for#save all customerID linked to the Account just inserted to the T_Loan select customer_ID from T_Account, T_Own where (ISSB=Account_BSB and AccountNo = Account_No) and (ISSB=new_Loan_AccountSB and AccountNo=new_Loan_AcctNo); declare continue handler for not found set v_finished=1;
                                             open owner_rec;
                                            repeat fetch owner_rec into temp_CID;
                                                        if(not v_finished) then
set temp_homeLoan =0;
set personalLoan_count=0;
set temp_toalLoanAnt = 0;
set temp_individualLoan = 0;
set temp_totalLoan = 0;
                                                              set temp_totalLoan = totalLoan(temp_CID)+1; #function used to calculate the total number of loans to a customerID set temp_homeLoan= numberOfHomeLoan(temp_CID); set personalLoan_count= numberOfPersonalLoan(temp_CID); set temp_totalLoanAm_ount(temp_CID)+new.LoanAmount;# function used to calculate the total Amount of all loans for a customerID set temp_totalLoanAmount to totalLoanAmount(temp_CID)+new.LoanAmount;# function used to calculate the total Amount of all loans for a customerID set temp_totalLoanAmount of individual loans to a customerID
                                                               set temp_IndividualLoan = CountOfIndividualAccount(temp_CID)+isIndividualAccount(new.Loan_AccountBSB, new.Loan_AcctNo);
                                                             if(new.Loan_Type='LT1') then
set temp_homeLoan= temp_homeLoan +1;
elseif (new.Loan_Type='LT3') finen
set personalLoan_count= personalLoan_count+1;
end if;
                                                             if (temp_totalLoan >8) then
set raiseError =1;
set msg = maximum number of loans is 8 ';
set v_finished=1;
elseif(temp_homeLoan>3) then
set raiseError =1;
set msg = maximum(3) of home loans is reached!';
set v_finished=1;
                                                               set v_lillsleu= |,
elseif (personalLoan_count>1) then
set raiseError = 1;
set msg = 'maximum(1) of personal loans is reached! ';
                                                                      set msg = 'maxi
set v_finished=
                                                              set v_inisned=1;
elseif (temp_totalLoanAmt>10000000.00) then
set raiseError =1;
set msg = 'adding new loan exceeds maximum loan amount $10000000.00';
set v_inished=!
selseif (temp_individualLoan>5) then
set raiseError =1;
set msg = 'maximum number of individual loan >5.
                                                               set msg = 'maximum number of individual loans is 5.'; set v_finished=1; end if;
                                                               set temp_CID=";
                                              until v_finished
                                             end repeat;
close owner_rec;
                                           if (raiseError=1) then
signal solstate '45000' set message_text =msg;
```

Function 1-TotalLoan

```
88
89
90
91
92
93
94
95
              delimiter //
              #function helps to the total number of loans for a customerID which tells you how many loans belong to that customerID
              drop function if exists totalLoan //
create function totalLoan(CID varchar(10))
                 returns int
                 deterministic
           p begin
96
97
98
99
100
101
102
103
104
                 declare result int default 0;
                  select count(*) into result
                 from T_Account, T_Own, T_Loan
                 where (BSB = Account_BSB and
                    AccountNo=Account_No and
                    BSB = Loan_AccountBSB and
AccountNo = Loan_AcctNo) and
Customer_ID=CID;
105
                 return result:
106
             end
07
108
              delimiter;
```

Function 2-numberOfHomeLoan

```
110
             delimiter //
111
             ##function help to find the number of homeLoans for CID
             drop function if exists numberOfHomeLoan //
112
113
             create function numberOfHomeLoan(CID varchar(11))
                returns int(11)
114
115
                deterministic
116
          begin
117
                declare result int default 0;
118
                select count(*) into result
from T_Account, T_Own, T_Loan
where (BSB = Account_BSB and
119
120
121
122
                        AccountNo=Account_No and
                        BSB = Loan_AccountBSB and
123
                        AccountNo = Loan_AcctNo) and (Loan_Type = 'LT1' and Customer_ID= CID);
124
125
126
127
                return result:
128
             end
129
130
             delimiter;
131
```

Function 3-numberOfPersonalLoan

```
132
           delimiter //
133
            #find out the number of personal loan
134
            drop function if exists numberOfPersonalLoan //
135
           create function numberOfPersonalLoan(CID varchar(11))
136
              returns int(11)
137
              deterministic
138
         p begin
139
              declare result int default 0;
              select count(*) into result
from T_Account, T_Own, T_Loan
140
141
                     where (BSB = Account_BSB and
142
143
                     AccountNo=Account_No and
                     BSB = Loan_AccountBSB and
144
                     AccountNo = Loan_AcctNo) and
145
146
                     (Loan_Type ='LT3' and Customer_ID= CID);
147
148
              return result:
149
           end
150
151
            delimiter;
152
```

Function 4 -totalLoanAmount

```
delimiter //
153
154
             #check the totalLoanAmount
             drop function if exists totalLoanAmount //
155
156
             create function totalLoanAmount(CID varchar(11))
                returns decimal(11,2)
157
158
               deterministic
159
          begin
160
                declare result int default 0;
               select sum(LoanAmount) into result
from T_Account, T_Own, T_Loan
where BSB = Account_BSB and
161
162
163
                  AccountNo=Account_No and
164
                  BSB = Loan_AccountBSB and
165
                  AccountNo = Loan_AcctNo and
166
                  Customer_ID=CID;
167
168
               return result:
169
             end
170
             delimiter;
171
172
173
```

Function 5-isIndividualAccount

```
#check if the account is an individualAccount or not
176
177
                  # yes return 1, no return 0 drop function if exists isIndividualAccount //
                  create function isIndividualAccount(loanBSB varchar(10), loanAcctNo varchar(10))
179
                      returns int
180
                       deterministic
181
182
              begin declare result int default 0
                      declare result int default 0;
select count(Customer_ID) into result
from T_Account, T_Own, T_Loan
where BSB = Account_BSB and
AccountNo=Account_No and
BSB = Loan_AccountBSB and
AccountNo = Loan_AcctNo and
BSB=loanBSB and AccountNo = loanAcctNo;
183
184
185
186
187
189
190
191
192
                      if(result=1) then
                          return 1;
193
                      end if;
194
                      return 0
195
                  end
196
                  delimiter :
197
```

Function 6-CountOfIndividualAccount

```
delimiter //

#this function returns the number of individaul account for a customerID
drop function if exists CountOfIndividualAccount (CID varchar(11))
RETURNS int(11)
DETERMINISTIC
DESCRATE FUNCTION CountOfIndividualAccount(CID varchar(11))
RETURNS int(11)
DETERMINISTIC
DESCRATE FUNCTION CountOfIndividualAccount(CID varchar(11))

### COUNTINE OF THE COUNTINE OF
```

iv. **Script for testing the trigger (and test results)**

```
# lest cases for homeLoan insertion

INSERT INTO T. Account VALUES (15,8810) "Acct1") # lest case for first homeloan for C1

INSERT INTO T. Account VALUES (15,000.00,111) BSB10 (Acct1) # lest case for first homeloan for C1

INSERT INTO T. Account VALUES (15,000.00,111) BSB10 (Acct1) # lest case for second homeloan for C1

INSERT INTO T. Account VALUES (15,100,000.00,111) BSB10 (Acct1) # lest case for second homeloan for C1

INSERT INTO T. Account VALUES (11,100,000.000,111) BSB11, "Acct1) # lest case for third homeloan for C1

INSERT INTO T. Account VALUES (11,100,000.000,111) BSB12, "Acct1) # lest into the homeloan for C1

INSERT INTO T. Account VALUES (12,100,500.000,111) BSB12, "Acct1) # lest into homeloan for C1

INSERT INTO T. Account VALUES (12,100,500.000,111) BSB12, "Acct1) # lest case for furth homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for firth homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for firth homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for firth homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for firth homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for firth individual loan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for sixth individual loan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for second personal loan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for second personal loan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for second homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for second homeloan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for eighith loan for C1

INSERT INTO T. Account VALUES (13,8812), "Acct1,"10.00,"AT3); # lest case for
```

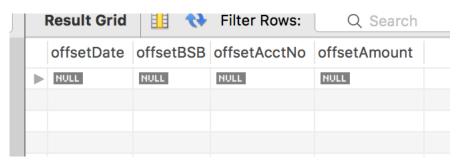
0	59	17:55:40	INSERT INTO T_Account VALUES ('BSB10','Acct1','10.00','AT3')	1 row(s) affected	0.0085 sec
0	60	17:55:40	INSERT INTO T_Own VALUES ('C1','BSB10','Acct1')	1 row(s) affected	0.0070 sec
0	61	17:55:40	INSERT INTO T_Loan VALUES ('L10','0.05','5000.00','LT1','BSB10','Acct1')	1 row(s) affected	0.019 sec
Ope	n a sc	ript file in th	s editor INTO T_Account VALUES ('BSB11','Acct1','10.00','AT3')	1 row(s) affected	0.0054 sec
0	63	17:55:40	INSERT INTO T_Own VALUES ('C1','BSB11','Acct1')	1 row(s) affected	0.012 sec
0	64	17:55:40	INSERT INTO T_Loan VALUES ('L11','0.05','5000.00','LT1','BSB11','Acct1')	1 row(s) affected	0.0072 sec
0	65	17:55:40	INSERT INTO T_Account VALUES ('BSB12','Acct1','10.00','AT3')	1 row(s) affected	0.0082 sec
0	66	17:55:40	INSERT INTO T_Own VALUES ('C1','BSB12','Acct1')	1 row(s) affected	0.0072 sec
0	67	17:55:40	INSERT INTO T_Loan VALUES ('L12','0.05','5000.00','LT1','BSB12','Acct1')	1 row(s) affected	0.0078 sec
0	68	17:55:43	INSERT INTO T_Account VALUES ('BSB13','Acct1','10.00','AT3')	1 row(s) affected	0.0086 sec
0	69	17:55:43	INSERT INTO T_Own VALUES ('C1','BSB13','Acct1')	1 row(s) affected	0.0071 sec
3	70	17:55:43	INSERT INTO T_Loan VALUES ('L13','0.05','5000.00','LT1','BSB13','Acct1')	Error Code: 1644. maximum(3) of home loans is reached!	0.0062 sec
0	71	17:55:48	INSERT INTO T_Account VALUES ('BSB20','Acct1','10.00','AT3')	1 row(s) affected	0.024 sec
0	72	17:55:48	INSERT INTO T_Own VALUES ('C1','BSB20','Acct1')	1 row(s) affected	0.014 sec
0	73	17:55:48	INSERT into T_Loan values ('L20','0.05','5000.00','LT2','BSB20', 'Acct1')	1 row(s) affected	0.026 sec
0	74	17:55:48	INSERT INTO T_Account VALUES ('BSB21','Acct1','10.00','AT3')	1 row(s) affected	0.028 sec
0	75	17:55:48	INSERT INTO T_Own VALUES ('C1','BSB21','Acct1')	1 row(s) affected	0.023 sec
8	76	17:55:48	INSERT into T_Loan values ('L21','0.05','5000.00','LT2','BSB20', 'Acct1')	Error Code: 1644. maximum number of individual loans is 5.	0.014 sec
0	77	17:55:52	INSERT INTO T_Account VALUES ('BSB14','Acct1','10.00','AT3')	1 row(s) affected	0.013 sec
0	78	17:55:53	INSERT INTO T_Own VALUES ('C1','BSB14','Acct1')	1 row(s) affected	0.0053 sec
8	79	17:55:53	INSERT into T_Loan values ('L14','0.05','5000.00','LT3','BSB14', 'Acct1')	Error Code: 1644. maximum(1) of personal loans is reached!	0.0052 sec
0	80	17:55:57	INSERT INTO T_Account VALUES ('BSB41','Acct1','10.00','AT3')	1 row(s) affected	0.0058 sec
0	81	17:55:57	INSERT INTO T_Own VALUES ('C2','BSB41','Acct1')	1 row(s) affected	0.0052 sec
8	82	17:55:57	INSERT INTO T_Loan VALUES ('L41','0.05','10000000.00','LT1','BSB41','Acct1')	Error Code: 1644. adding new loan exceeds maximum loan amoun	0.0046 sec
0	83	17:56:05	INSERT INTO T_Account VALUES ('BSB31','Acct1','10.00','AT3')	1 row(s) affected	0.0058 sec
0	84	17:56:05	INSERT INTO T_Own VALUES ('C1','BSB31','Acct1')	1 row(s) affected	0.0054 sec
0	85	17:56:05	INSERT INTO T_Own VALUES ('C2','BSB31','Acct1')	1 row(s) affected	0.0062 sec
0	86	17:56:05	INSERT into T_Loan values ('L31','0.05','5000.00','LT2','BSB31', 'Acct1')	1 row(s) affected	0.022 sec
0	87	17:56:09	INSERT INTO T_Account VALUES ('BSB32','Acct1','10.00','AT3')	1 row(s) affected	0.0076 sec
0	88	17:56:09	INSERT INTO T_Own VALUES ('C1','BSB32','Acct1')	1 row(s) affected	0.0094 sec
0	89	17:56:09	INSERT INTO T_Own VALUES ('C2','BSB32','Acct1')	1 row(s) affected	0.0052 sec
0	90	17:56:09	INSERT into T_Loan values ('L32','0.05','5000.00','LT2','BSB32', 'Acct1')	1 row(s) affected	0.012 sec
0	91	17:56:12	INSERT INTO T_Account VALUES ('BSB33','Acct1','10.00','AT3')	1 row(s) affected	0.0061 sec
0	92	17:56:12	INSERT INTO T_Own VALUES ('C1','BSB33','Acct1')	1 row(s) affected	0.0069 sec
0	93	17:56:12	INSERT INTO T_Own VALUES ('C2','BSB33','Acct1')	1 row(s) affected	0.0058 sec
3	94	17:56:12	INSERT into T_Loan values ('L33','0.05','5000.00','LT2','BSB33', 'Acct1')	Error Code: 1644. maximum number of loans is 8	0.0065 sec

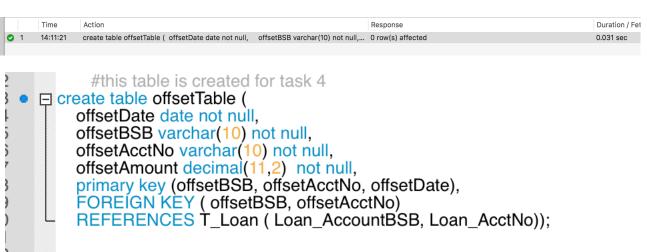
Assumption for task 3

- 1. I created loan accounts to test if the trigger works when insert new loan in T Loan.
- 2. New insertion is counted when determining if the insertion breaks 5 fives business rules as if the the customer already has 1 personal loan and if the loan type of new insertion for that customer is personal loan, the trigger should raise a error signal to reject the insertion.
- 3. The way I validate the trigger is using relevant test cases, successful insertion in for verifying previous business rule is needed for testing next business rule.
- 4. Assuming customers can create multiple accounts (including loan accounts) but, each account can only have maximum one loan.

Task 4 Interest Calculation

- 1. **Issues with the relational schema** no, because the offset table should be in a seperate table and is linked to the T_Loan. The offset is based on the individual loan account then there should be a relationship between offset table and T_account table to validate the constraint of the account detail as the account detail would be used to link both table together for foreign key integrity, because I assume the offset table is based on each individual account, so I need to modify the current schema for adding a table and link the offset table to T_account to constraint the integrity of account detail(BSB, accountNo).
- 2. **Suggested changes to the relational schema** As I mentioned in question 1, creating a offset table and linking to the T_account table is necessary in this situation as I want to ensure the data insertion is relevant to the loan account, and the foreign constraint allows me to validate this. There I would create an offset table using BSB
- 3. **Implementation of the suggested changes**





4. **Interest Calculation procedure**

(i) SQL Code

```
drop procedure if exists interestCalculator //
  3 4
                             create procedure interestCalculator (in startingDate date, in endDate date, in loanID varchar(11))
                                 declare not_found_row int default 0;
declare temp_offsetAmount decimal default 0;
declare temp_repaymentAmount decimal default 0;
                                 declare interest decimal(11,2);
declare tempBSB varchar(10);
  8
                                 declare tempAcctNo varchar(10);
declare interestRate decimal(10,8);
declare loanBalance decimal(11,2);
declare continue handler for not found set not_found_row =1;
10
11
12
13
14
15
16
17
                                  select BSB, AccountNo, LoanRate, AccountBal into tempBSB, tempAcctNo, interestRate, LoanBalance
                                 from T_Account, T_Loan
where BSB=Loan_AccountBSB and
AccountNo = Loan_AcctNo and
T_Loan.LoanID=loanID;
18
19
20
21
22
23
24
25
26
27
                                 set interest =0;
if(not_found_row) then
                                 signal sqlstate '45000' set message_text = 'no account found for that loanID'; end if;
                                  select subDate(endDate, interval 1 day) into endDate;
28
                                  set temp_offsetAmount =getOffset(endDate,tempBSB,tempAcctNo);
29
30
                                  while (datediff(endDate, startingDate)>0) do
                                 while (dateOff(ertiDate; startingDate) 30 of the post temp offsetAmount =getOffset(endDate,tempBSB,tempAcctNo); set interest = interest+ ((loanBalance+temp_offsetAmount)*interestRate/365); set loanBalance=loanBalance-getRepayment(endDate,loanID); select subDate(endDate, interval 1 day) into endDate; end while;
31
32
33
34
35
36
37
38
39
40
                                  select concat('Interest for ', loanID, ' is ',interest ) as 'result';
                           end
-//
delimiter;
41
42
43
44
45
                             delimiter //
                             definition if exists getOffset //
drop function if exists getOffset //
#get offset amount for a account and bsb and a date
create function getOffset(d date, bsb varchar(10), acct varchar(10))
46
47
48
                                 returns decimal(11,2)
deterministic
                                  beain
                                 declare row_not_found int default 0;
declare result decimal(11,2);
declare continue handler for not found set row_not_found =1;
49
50
51
52
53
54
55
56
57
58
                                  select offsetAmount into result
                                 from offsetTable where offsetBSB=bsb and
                                 offsetAcctNo = acct and
datediff(d,offsetDate)>=0
order by offsetDate desc
59
60
                                 if (row_not_found) then
61
62
63
64
65
66
67
68
                                 return 0;
else
                                 return result;
end if;
                                 end
//
                             delimiter ;
69
70
71
72
73
74
75
76
77
78
79
80
                            delimiter // #getRepayment for a data and loanID drop function if exists getRepayment // create function getRepayment(d date, loan varchar(10)) returns decimal(11,2)
                                  deterministic
                                  begin
                                 declare row_not_found int default 0;
declare result decimal(11,2);
declare continue handler for not found set row_not_found =1;
                                  select RepaymentAmount into result
81
82
83
                                  from T_Repayment where
                                 RepaymentDate=d and
Repayment_LoanID = loan;
84
85
                                  if (row_not_found) then
                                 return 0;
else
return result;
86
87
88
89
90
91
92
93
                                 end if:
                                 end
//
                             delimiter;
```

(ii) Errors/Warnings

errors would be shown if there is no Account found for the loanID

Task 4

Assumption

- I assume the days in a year is 365 as there is no written requirement for us to calculate the days of a year.
- 2. The parameters staring date and end date makes my procedure more flexible, as there is no requirements for not adding another parameters.
- 5. **Testing the procedure for monthly interest calculation**

```
2
3
4
5
6
7
8
9
             For task 4
              #Test case of simple calculator example for task 4
INSERT INTO T_Customer VALUES ('C4', 'matthew', 'lee', '1234567', 'dMail', '2015-10-12');
INSERT INTO T_Account VALUES ('BSB50', 'Acct1', '-60000.00', 'AT3');
INSERT INTO T_Own VALUES ('C4', 'BSB50', 'Acct1');
INSERT INTO T_Loan VALUES ('L50', '0.05', '670500.00', 'LT1', 'BSB50', 'Acct1');
              INSERT INTO T_Loan VALUES (L50, 0.03, 670500.00, LTT, BSB50, ACCTT);
INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
VALUES ('L50','5000','2018-10-20');
INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
VALUES ('L50','3000','2018-10-18');
12
13 •
14
15
16 •
              INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
17
                        VALUES ('L50', 5000', 2018-09-26');
18
19
20 • 21 • 22 • 23 • 24 •
              INSERT INTO offsetTable values ('2018-10-21', 'BSB50', 'Acct1', 7000);
               INSERT INTO offsetTable values ('2018-10-14', 'BSB50', 'Acct1', 5000);
              INSERT INTO offsetTable values ('2018-10-07','BSB50','Acct1',8000);
INSERT INTO offsetTable values ('2018-09-30','BSB50','Acct1',6000);
INSERT INTO offsetTable values ('2018-09-23','BSB50','Acct1',12000);
25
JE....
                    27
                 #March test case for task 4
                INSERT INTO T_Customer VALUES ('C5','Yvonne','Lam','12344','dMail','2015-10-12');
INSERT INTO T_Account VALUES ('BSB51','Acct1','-60000.00','AT3');
INSERT INTO T_Own VALUES ('C5','BSB51','Acct1');
INSERT INTO T_Loan VALUES ('L51','0.05','60000.00','LT1','BSB51','Acct1');
  28 •
  29 •
  30 •
  31 •
  32 •
                 INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
                 VALUES ('L51','6000','2018-03-20');
INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
  33
  34 •
  35
                          VALUES ('L51', '7000', '2018-03-18');
  36
  37 •
                 INSERT INTO T Repayment (Repayment LoanID, RepaymentAmount, RepaymentDate)
  38
                          VALUES ('L51', '5000', '2018-02-26');
  39
                INSERT INTO offsetTable values ('2018-03-21','BSB50','Acct1',6000); INSERT INTO offsetTable values ('2018-03-14','BSB50','Acct1',5000); INSERT INTO offsetTable values ('2018-03-07','BSB50','Acct1',8000); INSERT INTO offsetTable values ('2018-02-28','BSB50','Acct1',6000);
  40 •
  41 •
  42 •
  43 •
  44
```

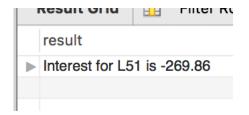
```
45
                      #May test case for task 4
                      INSÉRT INTO T_Customer VALUES ('C6','Vicky','Su','123444','dMail','2015-10-12');
46 •
47 •
                      INSERT INTO T_Account VALUES ('BSB52', 'Acct1', '-60000.00', 'AT3');
                      INSERT INTO T_Own VALUES ('C6','BSB52','Acct1');
INSERT INTO T_Loan VALUES ('L52','0.05','60000.00','LT1','BSB52','Acct1');
INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
VALUES ('L52','6000','2018-05-20');
48 •
49 •
50 •
51
                      INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
52 •
                                   VALUES ('L52', '7000', '2018-05-18');
53
54
                      INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
55 •
56
                                   VALUES ('L52', '5000', '2018-04-26');
57
                      INSERT INTO offsetTable values ('2018-05-21', 'BSB52', 'Acct1', 7000); INSERT INTO offsetTable values ('2018-05-14', 'BSB52', 'Acct1', 5000);
58 •
59 •
                      INSERT INTO offsetTable values ('2018-04-26', 'BSB50', 'Acct1', 12000);
60 •
61
 734
                            #August test case for task 4
                           INSERT INTO T_Customer VALUES ('C7', 'ricky', 'Tam', '123444', 'dMail', '2015-10-12');
INSERT INTO T_Account VALUES ('BSB53', 'Acct1', '-60000.00', 'AT3');
INSERT INTO T_Own VALUES ('C7', 'BSB53', 'Acct1');
INSERT INTO T_Danayment ('Denomination of the content 
  735 •
  736 •
  737 •
  738 •
  739 •
                            INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
                                         VALUES ('L53','6000','2018-08-20');
  740
  741 •
                            INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
                            VALUES ('L53','7000','2018-08-18');
INSERT INTO T_Repayment (Repayment_LoanID, RepaymentAmount, RepaymentDate)
  742
  743 •
                                         VALUES ('L53', '5000', '2018-07-26');
  744
  745
                           INSERT INTO offsetTable values ('2018-08-21','BSB52','Acct1',6000); INSERT INTO offsetTable values ('2018-08-14','BSB52','Acct1',5000); INSERT INTO offsetTable values ('2018-07-26','BSB52','Acct1',6000);
  746
  747
  748 •
 749
Oυ
                                             call interestCalculator('2018-09-24','2018-10-25', 'L50');
81
                                            call interestCalculator("2018-02-24", "2018-03-25", "L51");
call interestCalculator("2018-04-24", "2018-05-25", "L52");
call interestCalculator("2018-07-24", "2018-08-25", "L53");
82
83
84
85
86
87
```

Result

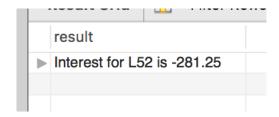
(i) On October 25, 2018



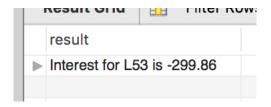
(ii) On March 25, 2018



(iii) On May 25, 2018



(iv) On August 25, 2018



Task 5 Transaction Management (Theory)

- a. A transaction is a logical unit of work on database, and it is an action or series of actions, carried out by application program or single user, that updates or reads the contents of the database. **Example** r1(T_Own) to read the all accounts relating to 'C1' in table T_Own is an action carried out by single user.
- **b.** Schedule is a sequence of operations carried out by multiple concurrent transactions, that preserves order of the operations in each individual transaction. This is different with transaction, as schedule can comprise of multiple transactions, each can have multiple actions where as transaction is a logical unit of work. **Example:** S1:(r1(T_Own);r2(T_Account);) involve two transactions-a transaction for reading T_Own and a transaction for reading T_Account.
- **c.** Conflict serialisable is that a schedule which:
 - 1. produces the same result as some serial execution
 - 2. Orders conflicting operations
 - 3. There are not conflicting operations in the schedule
- 4. Can be transformed into a serial schedule by swapping non-conflicting instructions Conflicting operations are :
 - two operations are from different transactions
 - operations are on same data item
 - at least one of the operations is a write operation

Conflicting operations might lead to lost update, dirty read, and unrepeated read A conflict serialisable schedule guarantee database consistency because:

If there are no conflicting operations in the schedule, we can freely change the order of the non-non-conflicting operations, we then can produce the equivalent serial schedule, which guarantee database consistency, if there are no conflicting operations. **Example** S1: r1(T_Own); r1(T_Account); w1(T_Repayment); r2(T_Loan); r3(T_Own); has no conflicting operations, so it is a conflicting serialised schedule.

d

Two schedules are regarded as conflicting equivalent if one schedule can be shown to be equivalent to the other by swapping non-conflicting operations, both schedules have the same set of transactions and the orders of conflicting actions are the same, then they leave the database in same state. Since the question says different conflicting serialisable schedules are made up of the same set of transactions, but the question doesn't say the orders of conflicting actions are the same, therefore, assuming the same set of transactions also includes same order of conflicting actions and same ordering of action within each transaction, then the two schedules are serialisable equivalent and they leave the database in the same state, as these two schedules can be proven to be conflict equivalent.

е.

View serialisability is that a schedule is view serialisable if it is view equivalent to a serial schedule. View serialisability allows blind write, therefore, it can protect the confidentiality of data, write data without exposing underlaying structure where as blind write is not allowable in conflict serialisable schedule. Blind write can also make the schedules are robust as reading before writing is not needed, which makes the transaction faster.

Example-view serialisable but not conflicting schedule

T1	T2	Т3
Read (T_Own)		
	Write(T_Own)	
Write(T_Own)		Write(T_Own)