Bank Management System

IT-252-MiniProject

1. Diptesh Banerjee(191ME298)

2. Akshay Bistagond(191EC203)

3. Abhishek Chavan(191MT001)

National Institute of Technology Karnataka

Bank Management System

A Database management system that maintains the day to day data that flows in a typical Bank

Contents

1. Problem Description

1.1 Motivation

Bank are a nondetachable part of our lives, our total personal economy depends on it. *As Manmohan Singh says*

> If you don't have a functioning financial system the world economy won't be revived. All the major economies have their responsibility to assist at a pace which is required to clean up the balance sheet of the banking system and to ensure that credit flows are resumed.

Technically speaking banking is an industry that handles cash, credit, and other financial transactions. Banks provide a safe place to store extra cash and credit. They offer savings accounts, certificates of deposit, and checking accounts. Banks use these deposits to make loans.

In the recent years, computers are included in almost all kind of works and jobs everyone come across in the routine. The availability of the software's for almost every process or every system has taken the world in its top-gear and fastens the day-to-day life. So, we have tried our best to develop the software program for the Bank Management System where all the tasks to manage the bank system are performed easily and efficiently. It manages all the transactions like new account entry, deposit as well as withdraw entry, transaction of money for various processes, loan entry, managing bills cash or cheque, etc. Thus, above features of this software will save transaction time and therefore increase the efficiency of the system. Requirements definition and management is recognized as a necessary step in the delivery of successful system s and software projects, discipline is also required by standards, regulations, and quality improvement initiatives. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budge. The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment.

1.2 Objective

The objective of this project is to design an good, efficient and a centralised Database management system for a typical bank with multiple branches in different locations. We hope to create a flawless and user friendly database management system. This Database management system is made using mySQL.

The data base has several salient features some of which are:-

- This database all the fields required all the day to day queries required for a typical bank
- One centralised database can be used by all the branches so that its easy to transfer accounts from branch to branch
- This is also a really simple database hence it is very easy and efficient to work with.

1.3 Instructions

The following are the requirements for the Database:

- A Bank has multiple branches that are identified by the branch_id and has details like branch name,branch city and assets
- A Bank branch hosts accounts and lends loans and has employees working under it.
- A Bank account is identified by an account number and is owned by one or more customers.It keeps record of the account balance and also takes care of the overdraft.
- An employee works for a Branch and is identified by employee_id and have there associated details like name, title, phone etc. The employees also have a role of either a manager or a worker.
- A customer is identified by an unique customer_id and has other details like name and city associated with them.
- A customer borrows loan that is offered by the bank. Each loan is attached with a loan number and holds the information about amount.
- All the payments made under a loan are also recorded with date, payment number and amount.

2. ER Diagram

2.1 Actors

- Bank Employee
- Customer

2.2 Some Sample Quaries

1. A Customer can
   - Borrow Loan
   - Check his account
   - Check his transactions check his branch
   - check his remaining loan amount
2. An Employee can
   - check his branch
   - check list of customers
   - check pending transactions
   - check loan status

2.3 ER Diagram

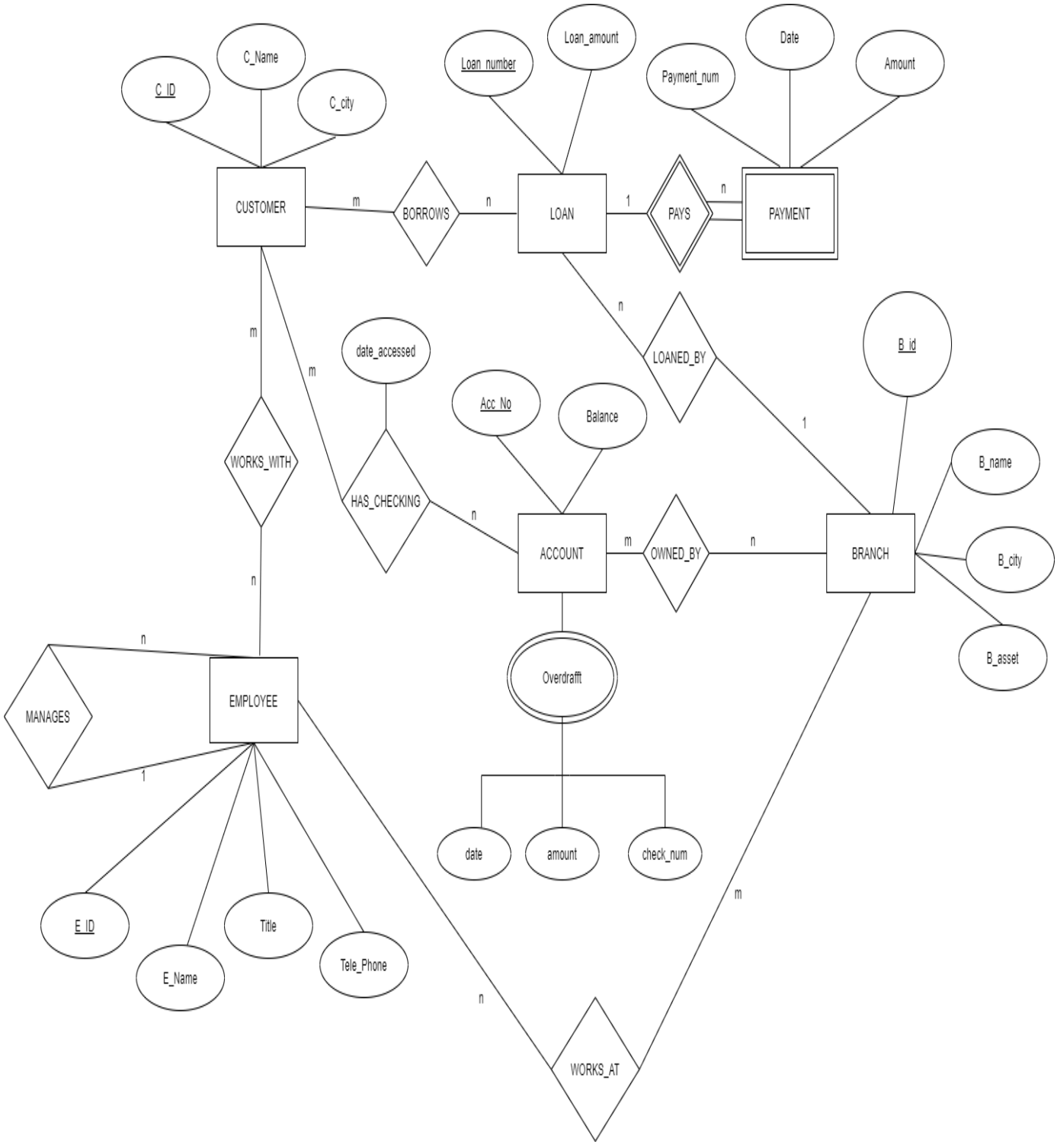

Figure 2.1: The ER Diagram

3. Global conceptual schema

3.1 Global Conceptual Schema

The transformation from the entity-relationship model to the relational model is very straightforward. A feasible set of relational schema is as follows.

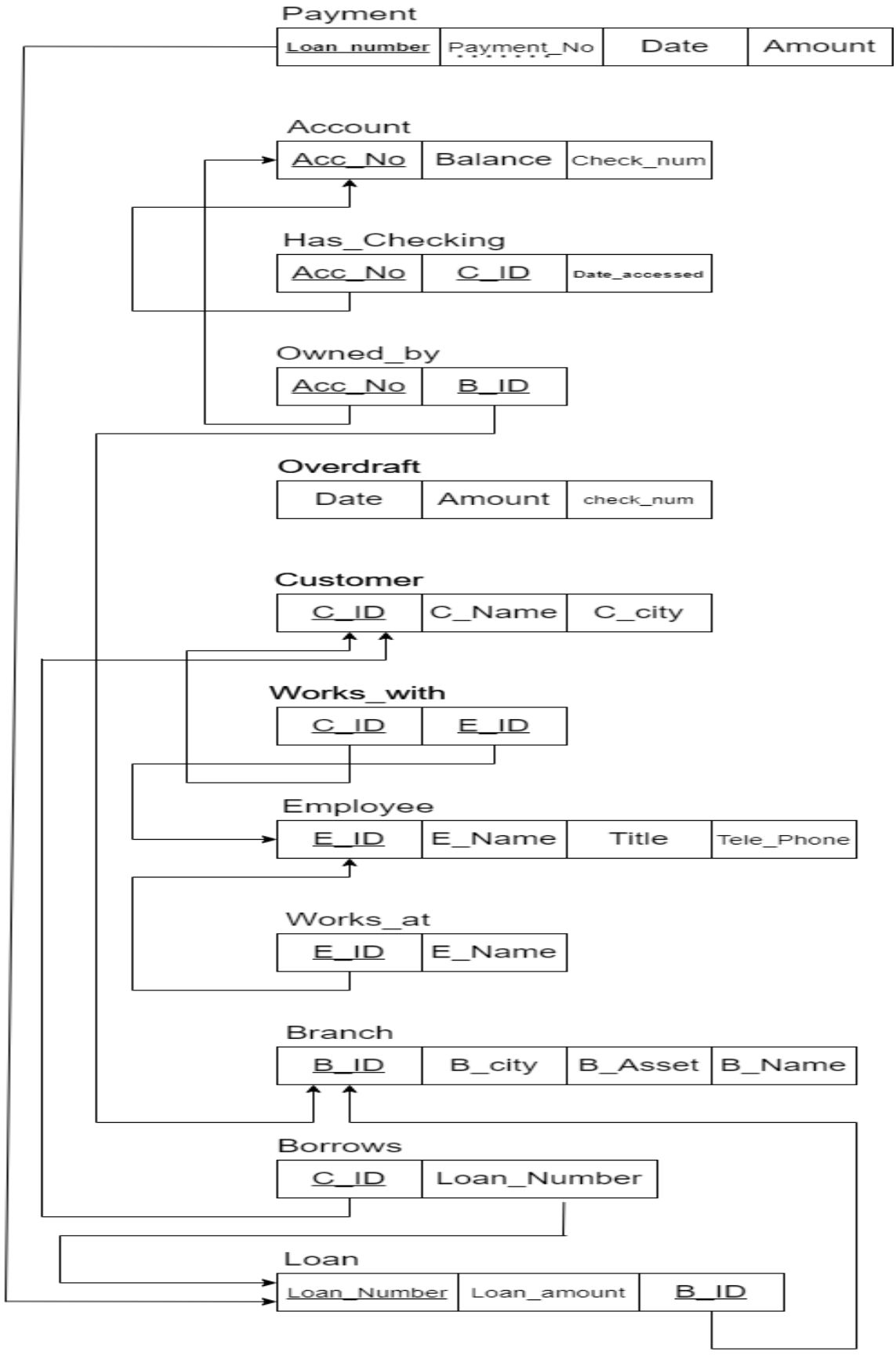
Payment
Loan_number
Payment_No
Date
Amount
Account
Acc_No
Balance
Check_num
Has_Checking
Acc_No
C_ID
Date_accessed
Owned_by
Acc_No
B_ID
Overdraft
Date
Amount
check_num
Customer
C_ID
C_Name
C_city
Works_with
C_ID
E_ID
Employee
E_ID
E_Name
Title
Tele_Phone
Works_at
E_ID
E_Name
Branch
B_ID
B_city
B_Asset
B_Name
Borrows
C_ID
Loan_Number
Loan
Loan_Number
Loan_amount
B_ID

Figure 3.1:

4. Normalization

1st Normal Form: A relation is said to be in first normal form then it should satisfy the following

- ✓ No multi-valued attribute
- ✓ No composite attribute
- ✓ identify primary key

Here, the relationship is converted to either relation or foreign key or merging relations.
Foreign key: giving primary key of one table as a reference to another table.

Payment:

Loan_number	Payment_No	Date	Amount

Account:

Acc_No	Balance

Has_checking:

Acc_No	C_ID

Owned_By:

Acc_No	B_ID

Customer:

C_ID	C_name	C_city

Employee:

E_ID	E_Name	Title	Tele_phone

Works_with:

C_ID	E_ID

Branch:

B_ID	B_city	B_Asset	B_name

Works_at:

E_ID	E_name

Loan:

Loan_number	Loan_amount	B_ID

Borrows:

C_ID	Loan_number

Note: Overdraft attribute is removed as it is multi-valued for 1st NF.

Outcome of 1st normalization:

- ✓ Primary key has been identified in each table using closure property (minimal super key)
- ✓ Composite attributes has been resolved
- ✓ Multi-valued attributes has been resolved

2nd Normal Form:

(i) Repeating column values are taken out and maintained in a separate table. So that change can be done only once in the new table rather than all entries in the first table. Rule is foreign key must be on the N side else again multi-value in a column will occur.

(ii) Identify **prime attribute** (part of candidate key that determines anything else), it is also called **partial dependency**, and eliminate it. Because, 2nd NF is based on **Full Functional dependency** (key should determine all other attributes in a table)

(iii) Use foreign key on many side

Payment:

Loan_number	Payment_No	Date	Amount

Account:

Acc_No	Balance

Has_checking:

Acc_No	C_ID

Owned_By:

Acc_No	B_ID

Customer:

C_ID	C_name	C_city

Employee:

E_ID	E_Name	Title	Tele_phone

Works_with:

C_ID	E_ID

Branch:

B_ID	B_city	B_Asset	B_name

Works_at:

E_ID	E_name

Loan:

Loan_number	Loan_amount	B_ID

Borrows:

C_ID	Loan_number

3rd Normal Form:

- ✓ Only columns with direct dependency of the primary key shall be in the entity.
- ✓ No transitive dependencies: non-prime attributes transitively depending on the key. . Example: A→B→C == A → C.
  A → BB is non-key attribute here
  B→C suddenly becomes key attribute here. because of this, we will get repeated values in a column. Therefore, it should be eliminated.
- ✓ 3rd NF should hold the condition that: if X → Y then
  Either X is a super key
  Or Y is a prime attribute
  Following this condition will never allow transitive dependency.

Payment:

Loan_number	Payment_No	Date	Amount

Account:

Acc_No	Balance

Has_checking:

Acc_No	C_ID

Owned_By:

Acc_No	B_ID

Customer:

C_ID	C_name	C_city

Employee:

E_ID	E_Name	Title	Tele_phone

Works_with:

C_ID	E_ID

Branch:

B_ID	B_city	B_Asset	B_name

Works_at:

E_ID	E_name

Loan:

Loan_number	Loan_amount	B_ID

Borrows:

C_ID	Loan_number

Every candidate key in a table determines all other attributes and no non-key attributes determine any attributes in the tables.

4th NF is not required since we eliminate repeated values in the 1st NF itself.

5. 20 Sample Queries

```
mysql> select a.cname,c.balance from customer a join haschecking b join account c on a.cid = b.c_id and c.accno = b.acc_
no;
+-------------------+---------+
| cname             | balance |
+-------------------+---------+
| Siddaram          |   98364 |
| Pundlik           |  274321 |
| Paul Jakob        |   50000 |
| Shekar Singh      |    8469 |
| Jeetain kattimani |   54284 |
| Somasid Rao       |   14590 |
| Kaarim saheb      | 5002734 |
| Pritam Dodmani    |    6402 |
| Sneha Patil       |  346364 |
| Lily Jack         |  314813 |
+-------------------+---------+
10 rows in set (0.00 sec)
```

Figure 5.1: Write a query to print the name and acc_balance of a customer

```
mysql> select a.cname as customerName,c.ename as employeeName from customer a join workswith b join employee c on a.cid
= b.c_id and c.eid = b.e_id;
+-------------------+--------------------+
| customerName      | employeeName       |
+-------------------+--------------------+
| Siddaram          | Kalyan Veerender   |
| Pundlik           | Jitendra Choudhary |
| Paul Jakob        | Jayadev Mitali     |
| Shekar Singh      | Hardeep Suksma     |
| Jeetain kattimani | Dhritiman Salim    |
| Somasid Rao       | Kalyan Veerender   |
| Kaarim saheb      | Jitendra Choudhary |
| Pritam Dodmani    | Jayadev Mitali     |
| Sneha Patil       | Hardeep Suksma     |
| Lily Jack         | Dhritiman Salim    |
+-------------------+--------------------+
10 rows in set (0.00 sec)
```

Figure 5.2: Write a query to print the customer name and employee name of the person handling customer

```
mysql> select a.ename as employeeName,c.bname as branchName from employee a join worksat b join branch c on a.eid = b.e_
id and c.bid = b.b_id;
+---------------------+--------------------+
| employeeName        | branchName         |
+---------------------+--------------------+
| Aditi Musunur       | Co-operative Bank  |
| Dharmadhrt Ramila   | Co-operative Bank  |
| Advitiya Sujeet     | Basaveshwara Bank  |
| Kalyan Veerender    | Basaveshwara Bank  |
| Jitendra Choudhary  | Basaveshwara Bank  |
| Devasru Subramanyan | Basaveshwara Bank  |
| Alagesan Poduri     | Kotiling Bank      |
| Jayadev Mitali      | Kotiling Bank      |
| Avidosa Vaisakhi    | Kotiling Bank      |
| Amrish Ilyasu       | siddeshwara Bank   |
| Hardeep Suksma      | siddeshwara Bank   |
| Barsati Sandipa     | siddeshwara Bank   |
| Vijai Sritharan     | Panchimsali Bank   |
| Dhritiman Salim     | Panchimsali Bank   |
| Avantas Ghosal      | Panchimsali Bank   |
+---------------------+--------------------+
15 rows in set (0.00 sec)
```

Figure 5.3: Write a query to print the employee name and branch in which he is working

```
mysql> select a.cname,d.bname from customer a join haschecking b join ownedby c join branch d on a.cid = b.c_id and b.ac
c_no = c.acc_no and c.b_id = d.bid;
+-------------------+-------------------+
| cname             | bname             |
+-------------------+-------------------+
| Siddaram          | Co-operative Bank |
| Somasid Rao       | Co-operative Bank |
| Pundlik           | Basaveshwara Bank |
| Kaarim saheb      | Basaveshwara Bank |
| Paul Jakob        | Kotiling Bank     |
| Pritam Dodmani    | Kotiling Bank     |
| Shekar Singh      | siddeshwara Bank  |
| Sneha Patil       | siddeshwara Bank  |
| Jeetain kattimani | Panchimsali Bank  |
| Lily Jack         | Panchimsali Bank  |
+-------------------+-------------------+
10 rows in set (0.00 sec)
```

Figure 5.4: write a query to print customer name and branch in which he has an account

```
mysql> select a.cname,c.loanamt from customer a join borrows b join loan c on a.cid = b.c_id and b.loan_no = c.loanno;
+--------------+---------+
| cname        | loanamt |
+--------------+---------+
| Paul Jakob   |   25000 |
| Shekar Singh |   40000 |
| Kaarim saheb |   10000 |
| Lily Jack    |  200000 |
+--------------+---------+
4 rows in set (0.00 sec)
```

Figure 5.5: Write a query to print customer name and the amount of loan he borrowed

```
mysql> select a.cname,c.date from customer a join borrows b join payment c on a.cid = b.c_id and b.loan_no = c.loan_no;
+--------------+------------+
| cname        | date       |
+--------------+------------+
| Paul Jakob   | 2019-12-22 |
| Shekar Singh | 2020-03-27 |
| Lily Jack    | 2020-04-07 |
+--------------+------------+
3 rows in set (0.01 sec)
```

Figure 5.6: Write a query to print customer name and last installment payment date

```
mysql> select a.cname,c.amt from customer a join borrows b join payment c on a.cid = b.c_id and b.loan_no = c.loan_no;
+--------------+-------+
| cname        | amt   |
+--------------+-------+
| Paul Jakob   | 10000 |
| Shekar Singh |  5000 |
| Lily Jack    | 50000 |
+--------------+-------+
3 rows in set (0.00 sec)
```

Figure 5.7: Write a query to print customer name and last installment amount paid

```
mysql> select a.cname,d.bname from customer a join borrows b join loan c join branch d on a.cid = b.c_id and b.loan_no =
 c.loanno and c.b_id = d.bid;
+--------------+-------------------+
| cname        | bname             |
+--------------+-------------------+
| Paul Jakob   | Basaveshwara Bank |
| Shekar Singh | siddeshwara Bank  |
| Kaarim saheb | Kotiling Bank     |
| Lily Jack    | Co-operative Bank |
+--------------+-------------------+
4 rows in set (0.00 sec)
```

Figure 5.8: Write a query to print customer name and branch name from which loan was taken

```
mysql> select a.bname,count(b.acc_no) from branch a join ownedby b on a.bid = b.b_id group by a.bname;
+-------------------+-----------------+
| bname             | count(b.acc_no) |
+-------------------+-----------------+
| Co-operative Bank |               2 |
| Basaveshwara Bank |               2 |
| Kotiling Bank     |               2 |
| siddeshwara Bank  |               2 |
| Panchimsali Bank  |               2 |
+-------------------+-----------------+
5 rows in set (0.00 sec)
```

Figure 5.9: write a query to print branch name and no of accounts it holds

```
mysql> select a.bname,count(b.e_id) as no_of_employees from branch a join worksat b on a.bid = b.b_id group by a.bname;
+------------------+-----------------+
| bname            | no_of_employees |
+------------------+-----------------+
| Co-operative Bank |               2 |
| Basaveshwara Bank |               4 |
| Kotiling Bank     |               3 |
| siddeshwara Bank  |               3 |
| Panchimsali Bank  |               3 |
+------------------+-----------------+
5 rows in set (0.00 sec)
```

Figure 5.10: Write a query to print branch name and no of employees that work in it

```
mysql> select a.cname,c.balance from customer a join haschecking b join account c on a.cid = b.c_id and b.acc_no = c.acc
no where c.balance = (select max(balance) from account);
+-------------+---------+
| cname       | balance |
+-------------+---------+
| Kaarim saheb | 5002734 |
+-------------+---------+
1 row in set (0.00 sec)
```

Figure 5.11: Write a query to print customer name with highest account balance

```
mysql> select a.cname from customer a where not exists(select *from borrows b where a.cid = b.c_id);
+-------------------+
| cname             |
+-------------------+
| Siddaram          |
| Pundlik           |
| Jeetain kattimani |
| Somasid Rao       |
| Pritam Dodmani    |
| Sneha Patil       |
+-------------------+
6 rows in set (0.00 sec)
```

Figure 5.12: Find customers with an account but not a loan.

```
mysql> select a.cname from customer a where a.cid in(select c_id from borrows);
+--------------+
| cname        |
+--------------+
| Paul Jakob   |
| Shekar Singh |
| Kaarim saheb |
| Lily Jack    |
+--------------+
4 rows in set (0.00 sec)
```

Figure 5.13: Find customers with both an account and a loan.

```
mysql> select bname from branch where basset > some(select basset from branch where bcity=
'Garsangi');.00 sec)
+--------------------+
| bname              |
+--------------------+
| Co-operative Bank  |
| Kotiling Bank      |
| Panchimsali Bank   |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

Figure 5.14: Find all banks with assets greater than at least one bank in Garsangi.

```
mysql> select max(total_bal) as largest_total from (select a.bname,sum(c.balance) as total
_bal from branch a join ownedby b join account c on c.accno = b.acc_no and b.b_id = a.bid
group by a.bname) as totals (bname,total_bal);
+---------------+
| largest_total |
+---------------+
|       5277055 |
+---------------+
1 row in set (0.00 sec)
```

Figure 5.15: Find the largest total account balance of any branch.

```
mysql> select a.bname,b.loanamt from branch a join loan b on a.bid = b.b_id where b.loanam
t = (select max(loanamt) from loan);
+------------------+---------+
| bname            | loanamt |
+------------------+---------+
| Co-operative Bank |  200000 |
+------------------+---------+
1 row in set (0.00 sec)
```

Figure 5.16: Find the bank with highest amount of given loan

```
mysql> select title as Designations, count(ename) as no_of_employees from employee group b
y title;
+--------------+-----------------+
| Designations | no_of_employees |
+--------------+-----------------+
| Manager      |               5 |
| Accountant   |               5 |
| Cashier      |               5 |
+--------------+-----------------+
3 rows in set (0.00 sec)
```

Figure 5.17: Find no of employees grouped by designations in each bank

```
mysql> select a.b_id from loan a where a.loanno in(select b.loan_no from payment b join lo
an a on a.loanno=b.loan_no and date > 2020-01-01);
+------+
| b_id |
+------+
|    2 |
|    4 |
|    1 |
+------+
3 rows in set, 1 warning (0.00 sec)
```

Figure 5.18: Find the branch ids where there has been payment after 01-01-2020

```
mysql> select a.ename from employee a where a.title = 'cashier' and a.eid in (select b.e_id from worksat b join branch c on b.b_id = c.
bid where c.bid in (select bid from branch where bcity='Kamatagi'));
+---------------+
| ename         |
+---------------+
| Avantas Ghosal |
+---------------+
1 row in set (0.00 sec)
```

Figure 5.19: Find the cashier who works in kamatagi

```
mysql> select eid,ename from employee where eid in (select e_id from worksat where b_id =
(select b_id from worksat where e_id=(select eid from employee where ename='Avantas Ghosal
')));
+-----+-----------------+
| eid | ename           |
+-----+-----------------+
|   5 | Vijai Sritharan |
|  10 | Dhritiman Salim |
|  15 | Avantas Ghosal  |
+-----+-----------------+
3 rows in set (0.00 sec)
```

Figure 5.20: Print name, id of all the colleagues of Avantas Ghosal

6. Views

```
mysql> create view ce as select a.cname,c.ename,c.title,c.telno from customer a join workswith b join employee c on a.ci
d = b.c_id and b.e_id = c.eid;
Query OK, 0 rows affected (0.01 sec)

mysql> select *from ce;
+------------------+--------------------+------------+--------+
| cname            | ename              | title      | telno  |
+------------------+--------------------+------------+--------+
| Siddaram         | Kalyan Veerender   | Accountant | 987622 |
| Somasid Rao      | Kalyan Veerender   | Accountant | 987622 |
| Pundlik          | Jitendra Choudhary | Accountant | 987654 |
| Kaarim saheb     | Jitendra Choudhary | Accountant | 987654 |
| Paul Jakob       | Jayadev Mitali     | Accountant | 765432 |
| Pritam Dodmani   | Jayadev Mitali     | Accountant | 765432 |
| Shekar Singh     | Hardeep Suksma     | Accountant | 678910 |
| Sneha Patil      | Hardeep Suksma     | Accountant | 678910 |
| Jeetain kattimani | Dhritiman Salim    | Accountant | 135791 |
| Lily Jack        | Dhritiman Salim    | Accountant | 135791 |
+------------------+--------------------+------------+--------+
10 rows in set (0.00 sec)
```

Figure 6.1: Create view to show customer, the employee's name, designation and telephone No. handling the customer

```
mysql> create view rich as select c.accno,a.cname,a.ccity,c.balance from customer a join haschecking b join account c on
 c.accno = b.acc_no and a.cid = b.c_id where c.balance > 50000;
Query OK, 0 rows affected (0.01 sec)

mysql> select *from rich;
+-------+------------------+----------+---------+
| accno | cname            | ccity    | balance |
+-------+------------------+----------+---------+
|  2001 | Siddaram         | Raipur   |   98364 |
|  2002 | Pundlik          | Chikodi  |  274321 |
|  2005 | Jeetain kattimani | Kamatagi |   54284 |
|  2007 | Kaarim saheb     | Chikodi  | 5002734 |
|  2009 | Sneha Patil      | Garsangi |  346364 |
|  2010 | Lily Jack        | Kamatagi |  314813 |
+-------+------------------+----------+---------+
6 rows in set (0.00 sec)
```

Figure 6.2: Create view to show account number, customer name, customer branch and balance

```
mysql> create view cb as select a.cname,a.ccity,d.bname,d.bcity from customer a join haschecking b join ownedby c join b
ranch d on a.cid = b.c_id and b.acc_no = c.acc_no and c.b_id = d.bid;
Query OK, 0 rows affected (0.00 sec)

mysql> select *from cb;
+------------------+----------+------------------+----------+
| cname            | ccity    | bname            | bcity    |
+------------------+----------+------------------+----------+
| Siddaram         | Raipur   | Co-operative Bank | Raipur   |
| Somasid Rao      | Raipur   | Co-operative Bank | Raipur   |
| Pundlik          | Chikodi  | Basaveshwara Bank | Chikodi  |
| Kaarim saheb     | Chikodi  | Basaveshwara Bank | Chikodi  |
| Paul Jakob       | Gulbarga | Kotiling Bank     | Gulbarga |
| Pritam Dodmani   | Gulbarga | Kotiling Bank     | Gulbarga |
| Shekar Singh     | Garsangi | siddeshwara Bank  | Garsangi |
| Sneha Patil      | Garsangi | siddeshwara Bank  | Garsangi |
| Jeetain kattimani | Kamatagi | Panchimsali Bank  | Kamatagi |
| Lily Jack        | Kamatagi | Panchimsali Bank  | Kamatagi |
+------------------+----------+------------------+----------+
10 rows in set (0.00 sec)
```

Figure 6.3: Create view to show customer name and bank name

```
mysql> create view b_avg as select a.bname,avg(c.balance) from branch a join ownedby b join account c on a.bid = b.b_id
and c.accno = b.acc_no group by a.bname;
Query OK, 0 rows affected (0.00 sec)

mysql> select *from b_avg;
+------------------+----------------+
| bname            | avg(c.balance) |
+------------------+----------------+
| Co-operative Bank |     56477.0000 |
| Basaveshwara Bank |   2638527.5000 |
| Kotiling Bank     |     28201.0000 |
| siddeshwara Bank  |    177416.5000 |
| Panchimsali Bank  |    184548.5000 |
+------------------+----------------+
5 rows in set (0.00 sec)
```

Figure 6.4: create a view to show bank name and average account balance per bank

```
mysql> create view remain as select a.cname,b.loanno,b.loanamt as total_loan,(b.loanamt-d.amt) as remaining_amount from
customer a join loan b join borrows c join payment d on a.cid = c.c_id and c.loan_no = b.loanno and c.loan_no = d.loan_n
o;
Query OK, 0 rows affected (0.01 sec)

mysql> select *from remain;
+--------------+--------+------------+-----------------+
| cname        | loanno | total_loan | remaining_amount |
+--------------+--------+------------+-----------------+
| Paul Jakob   |  60001 |      25000 |           15000 |
| Shekar Singh |  60002 |      40000 |           35000 |
| Lily Jack    |  60004 |     200000 |          150000 |
+--------------+--------+------------+-----------------+
3 rows in set (0.00 sec)
```

Figure 6.5: create a view to show people who have taken loan,their principal amount, and remaining loan amount

7. Storage Functions

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE GetCustomerLevel( IN acc_no INT, OUT lvl VARCHAR(20))
    -> BEGIN
    -> DECLARE credit DECIMAL(10,2) DEFAULT 0;
    -> SELECT balance INTO credit FROM account WHERE accno = acc_no;
    -> IF credit > 0 and credit < 311718.05 THEN SET lvl = 'IRON';
    -> END IF;
    -> IF credit >= 311718.05 and credit < 617034.10 THEN SET lvl = 'SILVER';
    -> END IF;
    -> IF credit >= 617034.10 and credit < 2809884.05 THEN SET lvl = 'GOLD';
    -> END IF;
IF cre    -> IF credit >= 2809884.05 THEN SET lvl = 'PLATINUM';
    -> END IF;
    -> END$$
LIMITER ;Query OK, 0 rows affected (0.01 sec)

mysql> CALL GetCustomerLevel(2009, @lvl);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @lvl;
+--------+
| @lvl   |
+--------+
| SILVER |
+--------+
1 row in set (0.00 sec)

mysql> CALL GetCustomerLevel(2007, @lvl);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @lvl;
+----------+
| @lvl     |
+----------+
| PLATINUM |
+----------+
1 row in set (0.00 sec)
```

Figure 7.1: Create a storage function to rate employees into iron,bronze, silver, gold and platinum based on their account balance

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE SP_ACCOUNT_EXISTS(IN acc_no INT,OUT state VARCHAR(100))
    -> BEGIN
    -> DECLARE temp INT default 0;
    -> SELECT count(accno) INTO temp FROM account WHERE accno = acc_no;
    -> IF temp = 1 THEN SET state = "Account present";
    -> END IF;
    -> IF temp = 0 THEN SET state = "No Account Present";
    -> END IF;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> CALL SP_ACCOUNT_EXISTS(2009, @state);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @state;
+------------------+
| @state           |
+------------------+
| Account present |
+------------------+
1 row in set (0.00 sec)

mysql> CALL SP_ACCOUNT_EXISTS(2011, @state);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @state;
+---------------------+
| @state              |
+---------------------+
| No Account Present |
+---------------------+
1 row in set (0.00 sec)
```

Figure 7.2: Create a storage function to determine if account is present or not

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE balance_check(IN acc_no INT,OUT balance DECIMAL(10,2))
    -> BEGIN
    -> SELECT c.balance INTO balance from customer a join haschecking b join account c on a.cid = b.c_id and c.accno = b.acc_no WHERE c
.accno = acc_no;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> CALL balance_check(2003, @balance);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @balance;
+----------+
| @balance |
+----------+
| 50000.00 |
+----------+
1 row in set (0.00 sec)
```

Figure 7.3: Create a storage function to determine account balance

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE emp_count(IN b_name VARCHAR(100),OUT countt INT)
    -> BEGIN
    -> SELECT count(b.e_id) INTO countt from branch a join worksat b on a.bid = b.b_id WHERE a.bname = b_name group by a.bname;
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> CALL emp_count("Basaveshwara Bank", @countt);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @countt;
+---------+
| @countt |
+---------+
|       4 |
+---------+
1 row in set (0.00 sec)
```

Figure 7.4: Create a storage function to determine accounts per bank

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE emp_search(IN designation VARCHAR(100),IN branch VARCHAR(100),OUT name VARCHAR(100))
    -> BEGIN
    -> SELECT a.ename INTO name from employee a where a.title = designation and a.eid in (SELECT b.e_id from worksat b join branch c on
 b.b_id = c.bid where c.bid in (SELECT bid from branch where bcity = branch));
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> CALL emp_search("Accountant","Gulbarga", @name);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT @name;
+----------------+
| @name          |
+----------------+
| Jayadev Mitali |
+----------------+
1 row in set (0.00 sec)
```

Figure 7.5: Create a storage function to search an employee based on branch and designation

8. Storage Procedures

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE NEW_ACC(IN
    -> gaccno INT,
    -> initial_deposit INT,
    -> customer INT,
    -> branch INT)
    -> BEGIN
    -> INSERT INTO account (accno, balance) VALUES (gaccno, initial_deposit);
    -> INSERT INTO haschecking (acc_no, c_id) VALUES (gaccno, customer);
    -> INSERT INTO ownedby (acc_no, b_id) VALUES (gaccno, branch);
    -> END//
Query OK, 0 rows affected (0.25 sec)

mysql> DELIMITER ;
mysql> call new_acc(2011,50000,7,5);
Query OK, 1 row affected (0.33 sec)

mysql> select *from account;
+-------+---------+
| accno | balance |
+-------+---------+
|  2001 |   98364 |
|  2002 |  274321 |
|  2003 |   50000 |
|  2004 |    8469 |
|  2005 |   54284 |
|  2006 |   14590 |
|  2007 | 5002734 |
|  2008 |    6402 |
|  2009 |  346364 |
|  2010 |  314813 |
|  2011 |   50000 |
+-------+---------+
11 rows in set (0.00 sec)
```

Figure 8.1: Create a storage procedure to insert a new account

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE DEPOSIT(IN
    -> gaccno INT,
    -> deposit_amt INT)
    -> BEGIN
    ->  UPDATE account
    ->  SET balance = balance + deposit_amt
    ->  WHERE accno = gaccno;
    -> INSERT INTO transaction_history (acc_no, amt_deposited, amt_withdrew)
    -> VALUES (gaccno, deposit_amt, 0);
    -> SELECT * FROM transaction_history WHERE acc_no = gaccno;
    -> END//
Query OK, 0 rows affected (0.08 sec)

mysql> DELIMITER ;
mysql> call deposit(2004,25000);
+--------+---------------+--------------+
| acc_no | amt_deposited | amt_withdrew |
+--------+---------------+--------------+
|   2004 |         25000 |            0 |
+--------+---------------+--------------+
1 row in set (0.29 sec)

Query OK, 0 rows affected (0.29 sec)

mysql> select *from account;
+-------+---------+
| accno | balance |
+-------+---------+
|  2001 |   98364 |
|  2002 |  199321 |
|  2003 |   50000 |
|  2004 |   33469 |
|  2005 |   54284 |
|  2006 |   14590 |
|  2007 | 5002734 |
|  2008 |    6402 |
|  2009 |  346364 |
|  2010 |  314813 |
|  2011 |   50000 |
+-------+---------+
```

Figure 8.2: Create a storage procedure to deposit amount

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE WITHDRAW(IN
    -> gaccno INT,
    -> withdrew_amt INT)
    -> BEGIN
    ->  UPDATE account
    ->  SET balance = balance-withdrew_amt
    ->  WHERE accno = gaccno;
    -> INSERT INTO transaction_history (acc_no, amt_deposited, amt_withdrew)
    -> VALUES (gaccno,0, withdrew_amt);
    -> SELECT * FROM transaction_history WHERE acc_no = gaccno;
    -> END//
Query OK, 0 rows affected (0.24 sec)

mysql> DELIMITER ;
mysql> call withdraw(2002,75000);
+--------+---------------+--------------+
| acc_no | amt_deposited | amt_withdrew |
+--------+---------------+--------------+
|   2002 |             0 |        75000 |
+--------+---------------+--------------+
1 row in set (0.40 sec)

Query OK, 0 rows affected (0.41 sec)

mysql> select *from account;
+-------+---------+
| accno | balance |
+-------+---------+
|  2001 |   98364 |
|  2002 |  199321 |
|  2003 |   50000 |
|  2004 |    8469 |
|  2005 |   54284 |
|  2006 |   14590 |
|  2007 | 5002734 |
|  2008 |    6402 |
|  2009 |  346364 |
|  2010 |  314813 |
|  2011 |   50000 |
+-------+---------+
```

Figure 8.3: Create a storage procedure to withdraw amount

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE NEW_CUSTOMER(IN
    -> custno INT,
    -> name VARCHAR(20),
    -> city VARCHAR(20),
    -> empno INT)
    -> BEGIN
    -> INSERT INTO customer (cid, cname, ccity) VALUES (custno, name, city);
    -> INSERT INTO workswith(c_id, e_id) VALUES (custno , empno);
    -> END//
Query OK, 0 rows affected (0.12 sec)

mysql> DELIMITER ;
mysql> call new_customer(11,"Arya Chopda","Raipur",11);
Query OK, 1 row affected (0.17 sec)

mysql> select *from customer;
+-----+-------------------+----------+
| cid | cname             | ccity    |
+-----+-------------------+----------+
|   1 | Siddaram          | Raipur   |
|   2 | Pundlik           | Chikodi  |
|   3 | Paul Jakob        | Gulbarga |
|   4 | Shekar Singh      | Garsangi |
|   5 | Jeetain kattimani | Kamatagi |
|   6 | Somasid Rao       | Raipur   |
|   7 | Kaarim saheb      | Chikodi  |
|   8 | Pritam Dodmani    | Gulbarga |
|   9 | Sneha Patil       | Garsangi |
|  10 | Lily Jack         | Kamatagi |
|  11 | Arya Chopda       | Raipur   |
+-----+-------------------+----------+
11 rows in set (0.00 sec)
```

Figure 8.4: Create a storage procedure to insert new customer

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE NEW_EMPLOYEE(IN
    -> empno INT,
    -> name VARCHAR(20),
    -> position VARCHAR(20),
    -> telph INT,
    -> branch INT)
    -> BEGIN
    -> INSERT INTO employee (eid, ename, title, telno) VALUES (empno, name, position, telph);
    -> INSERT INTO worksat (e_id, b_id) VALUES (empno, branch);
    -> END//
Query OK, 0 rows affected (0.25 sec)

mysql> DELIMITER ;
mysql> call new_employee(16,"Harush Kengal","Cashier",623812,3);
Query OK, 1 row affected (0.20 sec)

mysql> select *from employee;
+-----+---------------------+------------+--------+
| eid | ename               | title      | telno  |
+-----+---------------------+------------+--------+
|   1 | Aditi Musunur       | Manager    | 123456 |
|   2 | Advitiya Sujeet     | Manager    | 122916 |
|   3 | Alagesan Poduri     | Manager    | 113223 |
|   4 | Amrish Ilyasu       | Manager    | 186752 |
|   5 | Vijai Sritharan     | Manager    | 936225 |
|   6 | Kalyan Veerender    | Accountant | 987622 |
|   7 | Jitendra Choudhary  | Accountant | 987654 |
|   8 | Jayadev Mitali      | Accountant | 765432 |
|   9 | Hardeep Suksma      | Accountant | 678910 |
|  10 | Dhritiman Salim     | Accountant | 135791 |
|  11 | Dharmadhrt Ramila   | Cashier    | 246801 |
|  12 | Devasru Subramanyan | Cashier    | 159378 |
|  13 | Avidosa Vaisakhi    | Cashier    | 951739 |
|  14 | Barsati Sandipa     | Cashier    | 377377 |
|  15 | Avantas Ghosal      | Cashier    | 778028 |
|  16 | Harush Kengal       | Cashier    | 623812 |
+-----+---------------------+------------+--------+
16 rows in set (0.03 sec)
```

Figure 8.5: Create a storage procedure to insert new employee

9. Triggers

```
MySQL  localhost:33060+ ssl  project_1  SQL > DELIMITER //
MySQL  localhost:33060+ ssl  project_1  SQL > CREATE TRIGGER c_trigger
                                           -> AFTER INSERT ON transaction_history
                                           -> FOR EACH ROW
                                           -> BEGIN
                                           -> UPDATE account SET balance = balance - amt_withdrew*0.015 where new.acc_no=accno;
                                           -> END//
Query OK, 0 rows affected (0.0850 sec)
MySQL  localhost:33060+ ssl  project_1  SQL >
MySQL  localhost:33060+ ssl  project_1  SQL > DELIMITER ;
```

Figure 9.1: Create a trigger to deduce 1.5% of the withdrawn amount as tax

```
MySQL  localhost:33060+ ssl  project_1  SQL > DROP TRIGGER IF EXISTS d_trigger;
Query OK, 0 rows affected, 1 warning (0.0372 sec)
Note (code 1360): Trigger does not exist
MySQL  localhost:33060+ ssl  project_1  SQL > DELIMITER //
MySQL  localhost:33060+ ssl  project_1  SQL > CREATE TRIGGER d_trigger
                                           -> AFTER INSERT ON transaction_history
                                           -> FOR EACH ROW
                                           -> BEGIN
                                           -> UPDATE account SET balance = balance + amt_deposited where new.acc_no=accno;
                                           -> END//
Query OK, 0 rows affected (0.0987 sec)
MySQL  localhost:33060+ ssl  project_1  SQL >
MySQL  localhost:33060+ ssl  project_1  SQL > DELIMITER ;
```

Figure 9.2: Create a trigger to deposit into account

```
MySQL  localhost:33060+ ssl  project_1  SQL > DELIMITER //
MySQL  localhost:33060+ ssl  project_1  SQL > CREATE TRIGGER a_trigger
                                           -> BEFORE DELETE ON transaction_history
                                           -> FOR EACH ROW
                                           -> BEGIN
                                           -> INSERT INTO transaction_history_archive (acc_no, amt_deposited, amt_withdrew)
                                           -> SELECT old.acc_no, old.amt_deposited, old.amt_withdrew;
                                           -> END//
Query OK, 0 rows affected (0.1978 sec)
MySQL  localhost:33060+ ssl  project_1  SQL > DELIMITER ;
```

Figure 9.3: Create a trigger to insert into transaction history before performing a delete operation