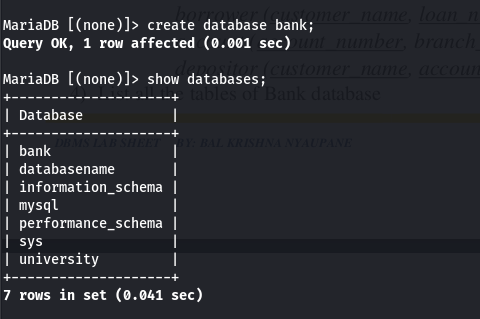
# DBMS QUERIES

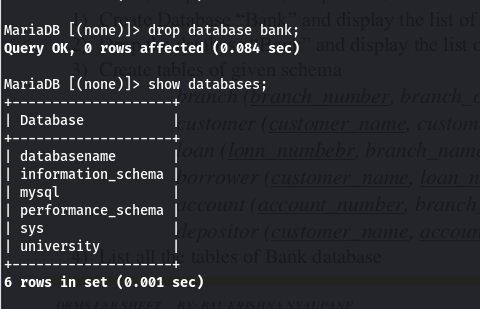
## 1) Create Database “Bank” and display the list of database

|  |
| --- |
| create database bank; show databases; |



**2) Drop database “Bank”**

|  |
| --- |
| drop database bank; |



## 3) Create tables of given schema

## - branch(branch\_number, branch\_name, branch\_city, assets)

## - customer(customer\_number,customer\_name, customer\_street, customer\_city)

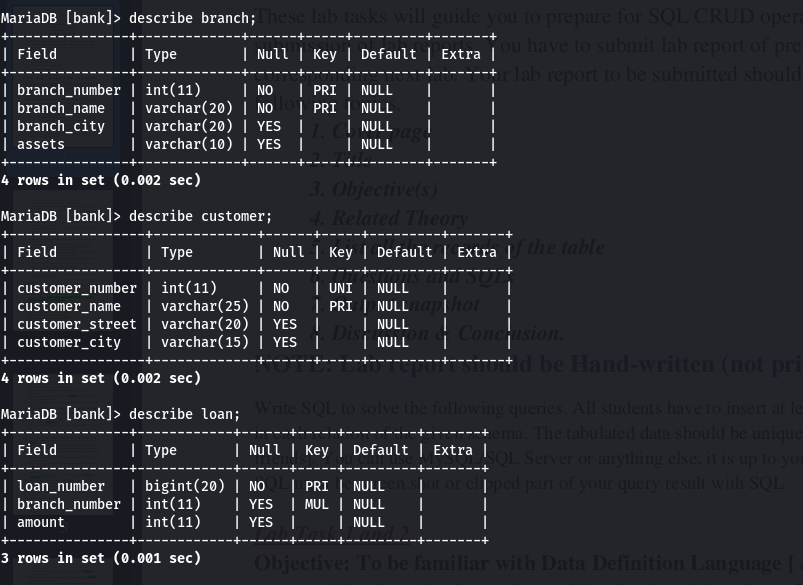
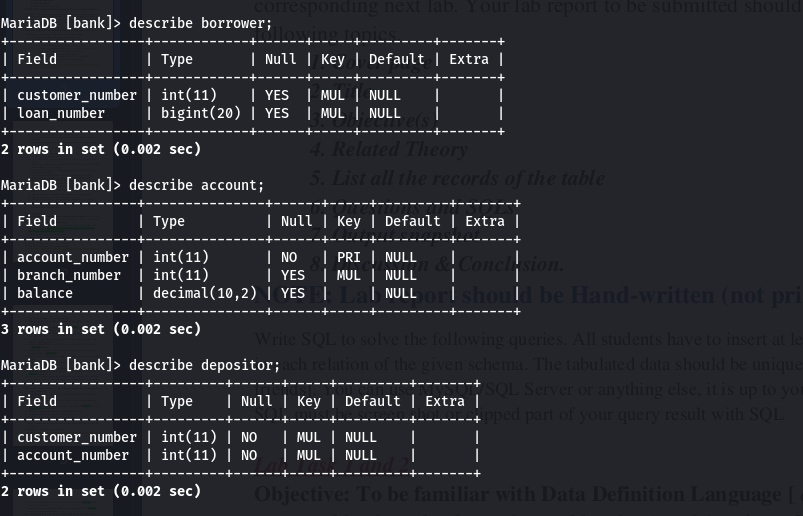
## - loan(loan\_number,branch\_number,amount)

## - borrower(customer\_number,loan\_number)

## - account(account\_number,branch\_number,balance)

## - depositor(customer\_number,account\_number)

|  |
| --- |
| -- Branch  create table branch( branch\_number int unique, branch\_name varchar(20), branch\_city varchar(20), assets varchar(10), primary key(branch\_number,branch\_name));  -- Customer create table customer( customer\_number int unique, customer\_name varchar(25), customer\_street varchar(20), customer\_city varchar(15), primary key(customer\_name, customer\_number));  -- Loan CREATE TABLE loan ( loan\_number bigint unique, branch\_number int, amount int, PRIMARY KEY (loan\_number), constraint fk\_const foreign key(branch\_number)references branch(branch\_number) on delete cascade on update cascade);  -- Borrower create table borrower( customer\_number int, loan\_number bigint, constraint fk\_const5 foreign key(customer\_number) references customer(customer\_number) on delete cascade on update cascade, constraint fk\_loan4 foreign key(loan\_number) references loan(loan\_number) on delete cascade on update cascade);  -- Account create table account( account\_number int unique, branch\_number int, balance decimal(10,2), primary key(account\_number), constraint fk\_cons foreign key(branch\_number)references branch(branch\_number) on delete cascade on update cascade);  -- Depositor create table depositor( customer\_number int not null,  account\_number int not null, constraint fk\_customer foreign key(customer\_number)references customer(customer\_number) on delete cascade on update cascade, constraint fk\_acc foreign key(account\_number) references account(account\_number)on delete cascade on update cascade); |

## 4) List all the tables

|  |
| --- |
| show tables; |

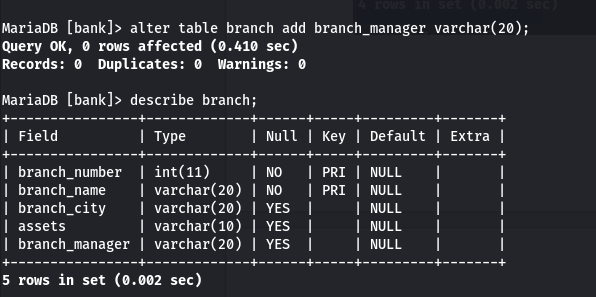


## 5) List all the columns of “branch” and “customer” tables

|  |
| --- |
| show columns from branch; describe customer; |

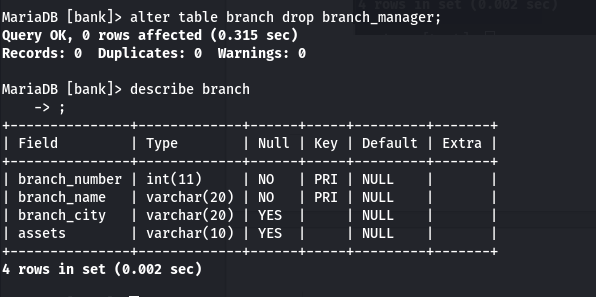
## 6) Add column “Branch\_manger” in branch table and list all the columns of branch table

|  |
| --- |
| alter table branch add branch\_manager varchar(20); show columns from branch; |



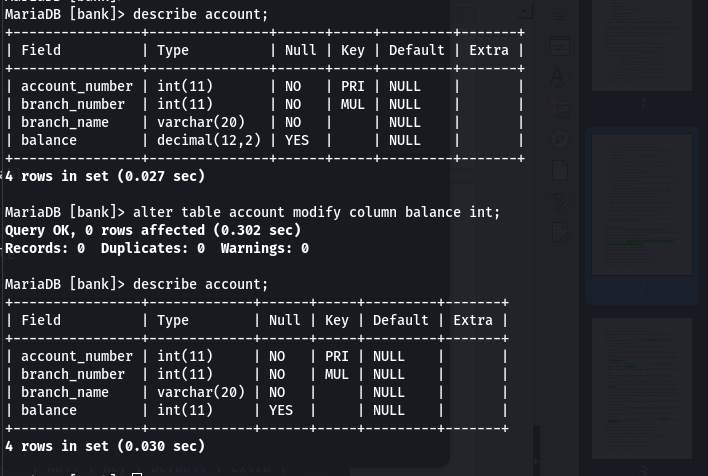
## 7) Drop column “Branch\_manger” from branch table and list all the columns of branch table

|  |
| --- |
| *alter table branch drop branch\_manager; describe branch;* |



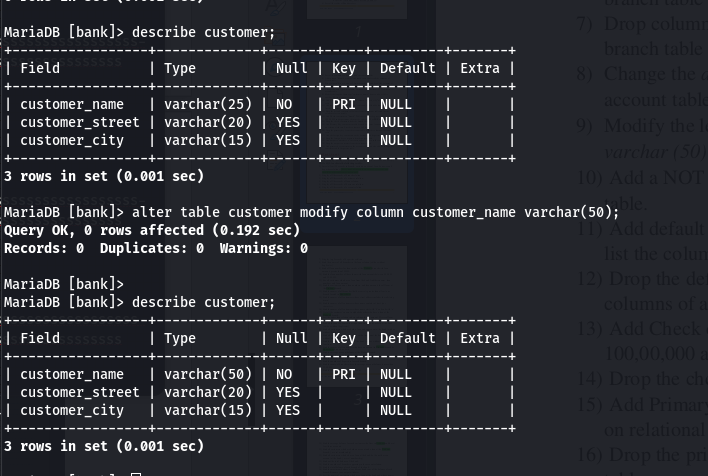
## 8) Change the data type of a column “balance” from decimal (12,2) to int in account table.

|  |
| --- |
| alter table account modify column balance int; describe account; |



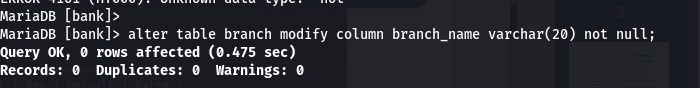
## 9) Modify the length of data type “customer\_name” from varchar (30) to varchar (50) in a customer table.

|  |
| --- |
| alter table customer modify column customer\_name varchar(50); |



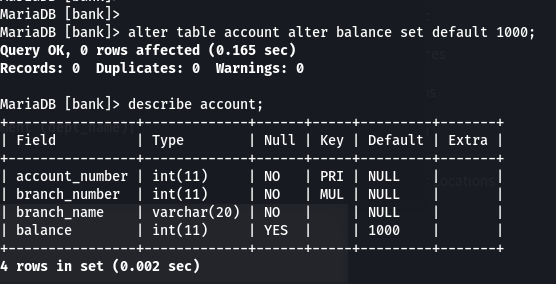
## 10) Add a NOT NULL constraint in the column “branch\_city” of the branch table.

|  |
| --- |
| alter table branch modify column branch\_city varchar(20) not null; |



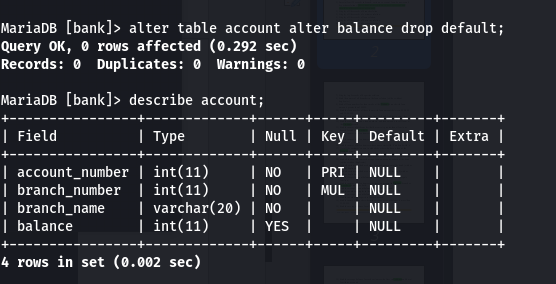
## 11) Add default constraint in “account” table and set default balance is 1000. And list the columns of account table.

|  |
| --- |
| alter table account alter balance set default 1000; |



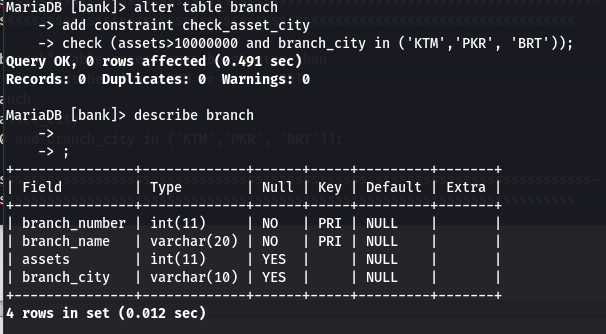
## 12) Drop the default constraint of balance from the account table and list the columns of account table.

|  |
| --- |
| alter table account alter balance drop default; |



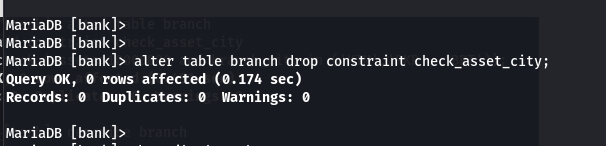
## 13) Add Check constraint in “branch” Table assets must be greater than 100,00,000 and department city must be either KTM or PKR or BRT.

|  |
| --- |
| alter table branch add constraint check\_asset\_city check (assets>10000000 and branch\_city in ('KTM','PKR', 'BRT')); |



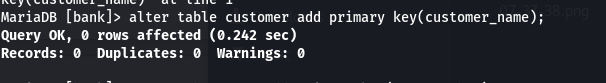
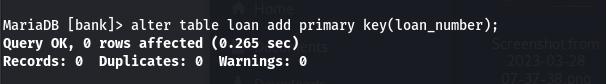
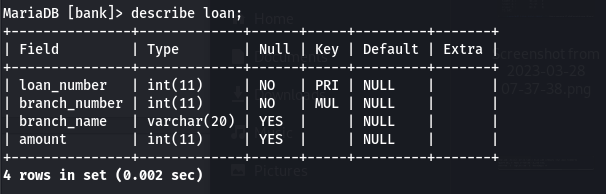
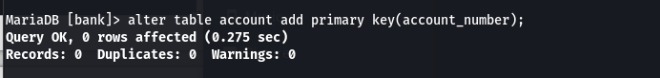
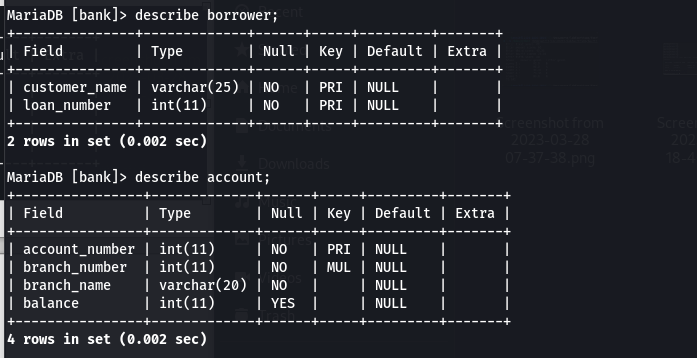
14) Drop the check constraints from the branch table.

|  |
| --- |
| *alter table branch drop constraint check\_asset\_city;* |



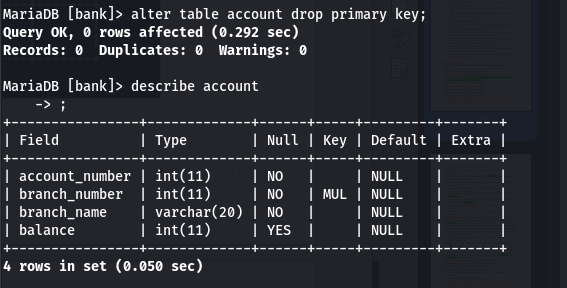
## 15) Add Primary Key constraint in table branch, customer, loan and account based on relational schema. List the columns of all tables.

|  |
| --- |
| alter table branch add primary key(branch\_number); alter table customer add primary key(customer\_name); alter table loan add primary key(loan\_number); alter table account add primary key(account\_number); |

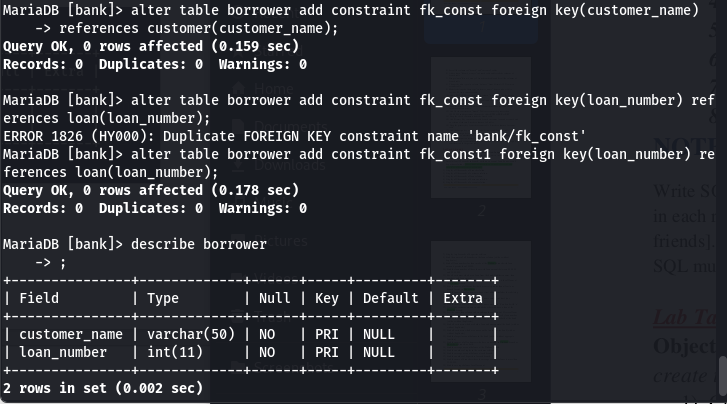
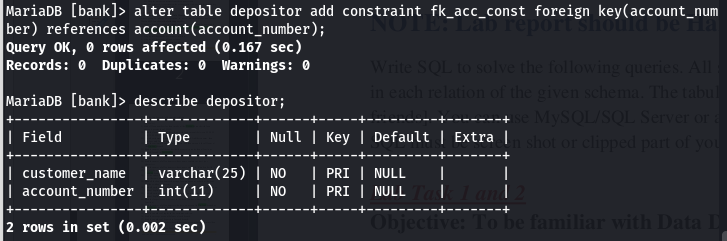
## 16) Drop the primary key from the account table. List the columns of account table.

|  |
| --- |
| alter table account drop primary key; |



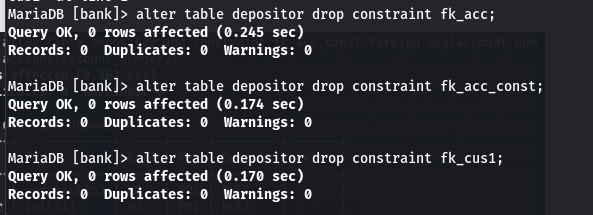
## 17) Add the foreign key constraints on the borrower and depositor table.

|  |
| --- |
| alter table borrower add constraint fk\_const foreign key(customer\_name) references customer(customer\_name); alter table borrower add constraint fk\_const1 foreign key(loan\_number) references loan(loan\_number); alter table depositor add constraint fk\_acc\_const foreign key(account\_number) references account(account\_number); alter table depositor add constraint cons foreign key(customer\_name) references customer(customer\_name); |

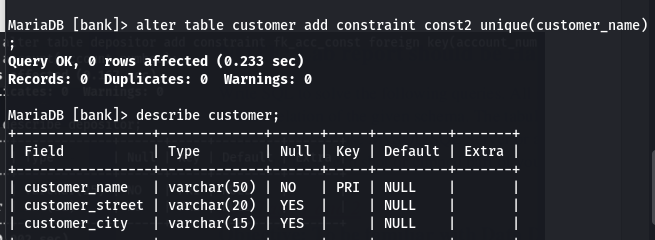
## 18) Drop the foreign key constraints form depositor table.

|  |
| --- |
| alter table depositor drop constraint fk\_acc; alter table depositor drop constraint fk\_acc\_const; alter table depositor drop constraint fk\_cus1; |



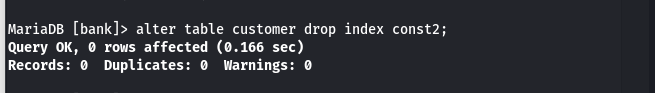
## 19) Add unique constraints on “customer\_name” column in the customer table.

|  |
| --- |
| alter table customer add constraint const2 unique(customer\_name); |



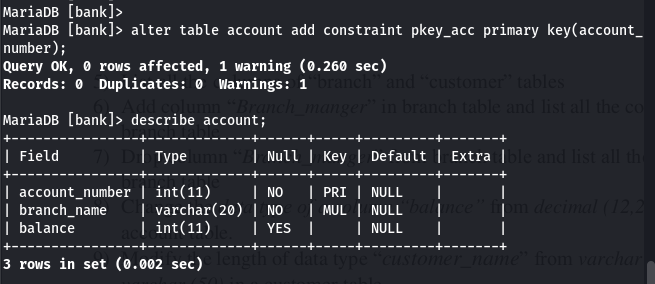
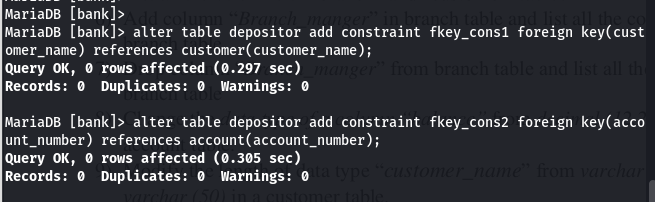
## 20)Drop the unique constraints from the customer table.

|  |
| --- |
| alter table customer drop index const2; |



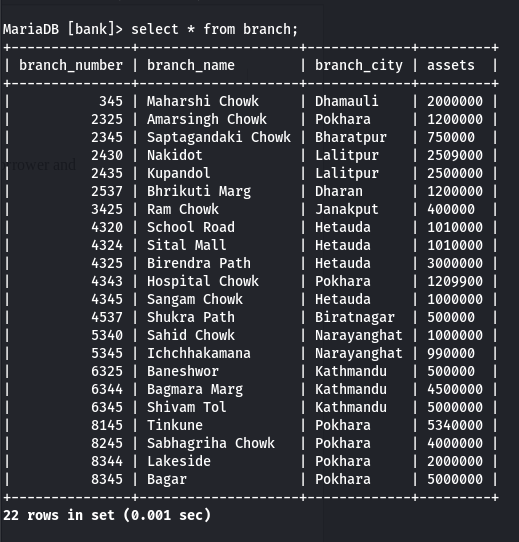
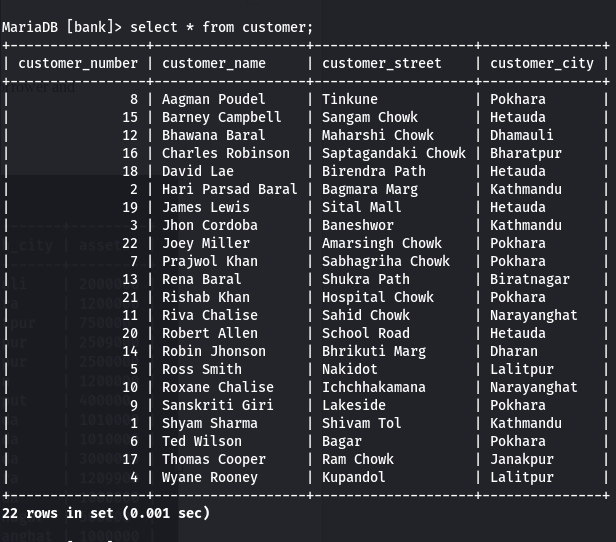
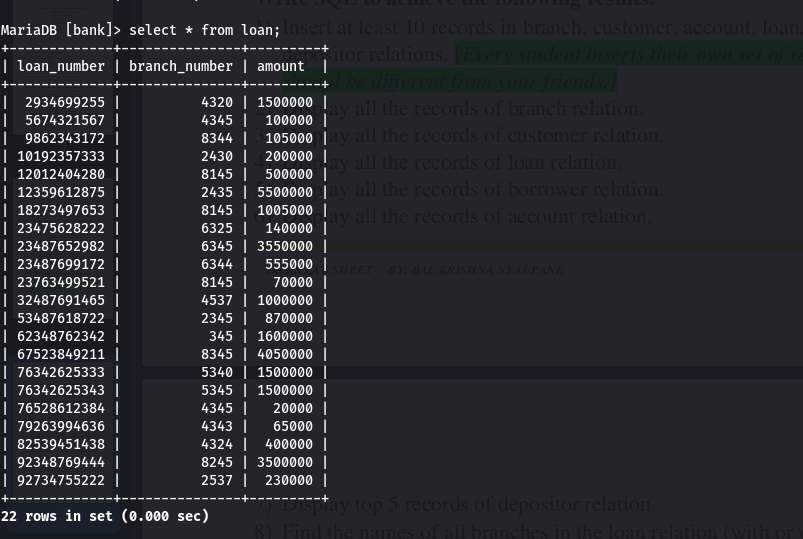
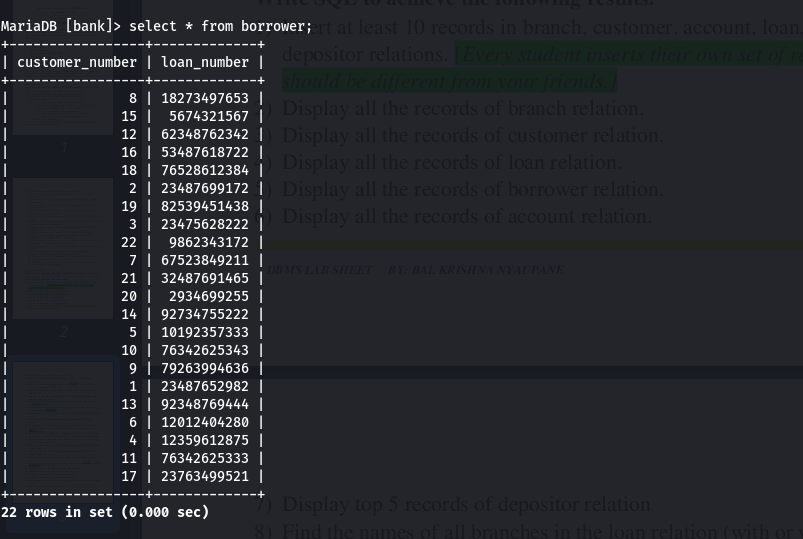
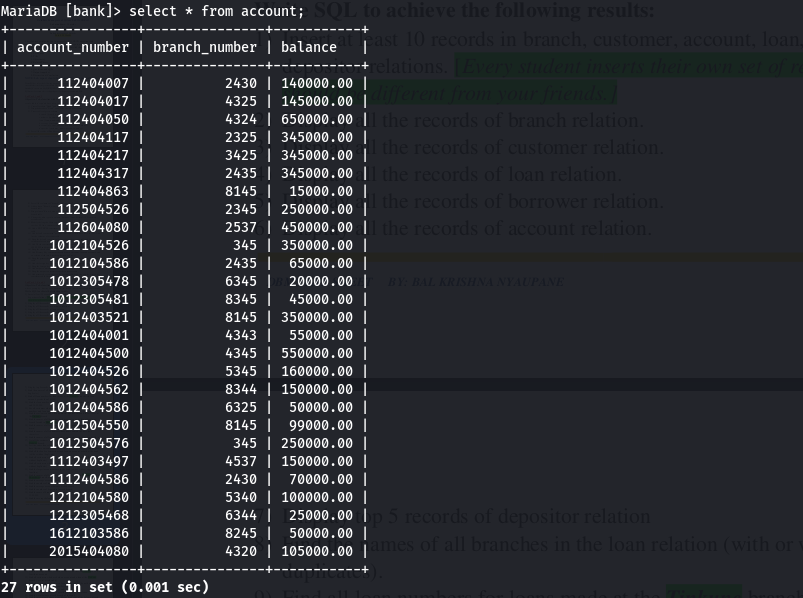
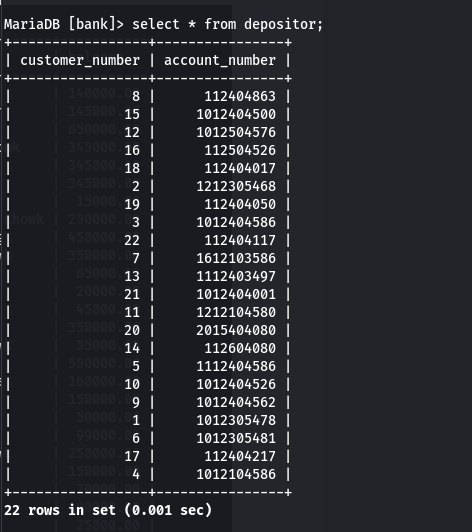
## 21) Add primary key constraint in the account table. And also add the foreign key constraints on the depositor table

|  |
| --- |
| alter table depositor add constraint fkey\_cons1 foreign key(customer\_name) references customer(customer\_name); alter table depositor add constraint fkey\_cons2 foreign key(account\_number) references account(account\_number); |

## 22) Insert at least 10 records in branch,customer, account, loan, borrower and depositor relations.

|  |
| --- |
| -- Branch *insert into branch values(6345,'Shivam Tol','Kathmandu',5000000); insert into branch values(6344,'Bagmara Marg','Kathmandu',4500000); insert into branch values(6325,'Baneshwor','Kathmandu',500000); insert into branch values(8345,'Bagar','Pokhara', 5000000); insert into branch values(8245,'Sabhagriha Chowk','Pokhara', 4000000); insert into branch values(8145,'Tinkune','Pokhara', 5340000); insert into branch values(8344,'Lakeside','Pokhara', 2000000); insert into branch values(5345,'Ichchhakamana','Narayanghat', 990000); insert into branch values(5340,'Sahid Chowk','Narayanghat', 1000000); insert into branch values(0345,'Maharshi Chowk','Dhamauli', 2000000); insert into branch values(4537,'Shukra Path','Biratnagar', 500000); insert into branch values(2537,'Bhrikuti Marg','Dharan',1200000); insert into branch values(4345,'Sangam Chowk','Hetauda',1000000); insert into branch values(2345,'Saptagandaki Chowk','Bharatpur',750000); insert into branch values(3425,'Ram Chowk','Janakput',400000); insert into branch values(2435,'Kupandol','Lalitpur',2500000); insert into branch values(2430,'Nakidot','Lalitpur',2509000); insert into branch values(4325,'Birendra Path','Hetauda',3000000); insert into branch values(4320,'School Road','Hetauda',1010000); insert into branch values(4324,'Sital Mall','Hetauda',1010000); insert into branch values(2325,'Amarsingh Chowk','Pokhara',1200000); insert into branch values(4343,'Hospital Chowk','Pokhara',1209900);* -- Customer *insert into customer values(1,'Shyam Sharma','Shivam Tol','Kathmandu'); insert into customer values(2,'Hari Parsad Baral','Bagmara Marg','Kathmandu'); insert into customer values(3,'Jhon Cordoba','Baneshwor','Kathmandu'); insert into customer values(4,'Wyane Rooney','Kupandol','Lalitpur'); insert into customer values(5,'Ross Smith','Nakidot','Lalitpur'); insert into customer values(6,'Ted Wilson','Bagar','Pokhara'); insert into customer values(7,'Prajwol Khan','Sabhagriha Chowk','Pokhara'); insert into customer values(8,'Aagman Poudel','Tinkune','Pokhara'); insert into customer values(9,'Sanskriti Giri','Lakeside','Pokhara'); insert into customer values(10,'Roxane Chalise','Ichchhakamana','Narayanghat'); insert into customer values(11,'Riva Chalise','Sahid Chowk','Narayanghat'); insert into customer values(12,'Bhawana Baral','Maharshi Chowk','Dhamauli'); insert into customer values(13,'Rena Baral','Shukra Path','Biratnagar'); insert into customer values(14,'Robin Jhonson','Bhrikuti Marg','Dharan'); insert into customer values(15,'Barney Campbell','Sangam Chowk','Hetauda'); insert into customer values(16,'Charles Robinson','Saptagandaki Chowk','Bharatpur'); insert into customer values(17,'Thomas Cooper','Ram Chowk','Janakpur'); insert into customer values(18,'David Lae','Birendra Path','Hetauda'); insert into customer values(19,'James Lewis','Sital Mall','Hetauda'); insert into customer values(20,'Robert Allen','School Road','Hetauda'); insert into customer values(21,'Rishab Khan','Hospital Chowk','Pokhara'); insert into customer values(22,'Joey Miller','Amarsingh Chowk','Pokhara'); insert into customer values(23,'Joy Miller','main road','Pokhara'); insert into customer values(24,'Burno Cordoba','New road','Pokhara');* -- Account *insert into account values(01012305478,6345,20000); insert into account values(01212305468,6344,25000); insert into account values(01012404586,6325,50000); insert into account values(01012104586,2435,65000); insert into account values(01112404586,2430,70000); insert into account values(01012305481,8345,45000); insert into account values(01612103586,8245,50000); insert into account values(00112404863,8145,15000); insert into account values(01012404562,8344,150000); insert into account values(01012404526,5345,160000); insert into account values(01212104580,5340,100000); insert into account values(01012504576,0345,250000); insert into account values(01112403497,4537,150000); insert into account values(0112504526,2345,250000); insert into account values(01012104526,0345,350000); insert into account values(00112604080,2537,450000); insert into account values(01012504550,8145,99000); insert into account values(01012404500,4345,550000); insert into account values(00112404050,4324,650000); insert into account values(01012403521,8145,350000); insert into account values(01012404001,4343,55000); insert into account values(02015404080,4320,105000); insert into account values(00112404007,2430,140000); insert into account values(00112404017,4325,145000); insert into account values(00112404117,2325,345000); insert into account values(00112404217,3425,345000); insert into account values(00112404317,2435,345000);* -- Loan *insert into loan values(05674321567,4345,100000); insert into loan values(76528612384,4345,20000); insert into loan values(10192357333,2430,200000); insert into loan values(12012404280,8145,500000); insert into loan values(79263994636,4343,65000); insert into loan values(23763499521,8145,70000); insert into loan values(82539451438,4324,400000); insert into loan values(92734755222,2537,230000); insert into loan values(09862343172,8344,105000); insert into loan values(02934699255,4320,1500000); insert into loan values(62348762342,0345,1600000); insert into loan values(32487691465,4537,1000000); insert into loan values(76342625343,5345,1500000); insert into loan values(76342625333,5340,1500000); insert into loan values(92348769444,8245,3500000); insert into loan values(67523849211,8345,4050000); insert into loan values(12359612875,2435,5500000); insert into loan values(23487652982,6345,3550000); insert into loan values(23487699172,6344,555000); insert into loan values(18273497653,8145,1005000); insert into loan values(23475628222,6325,140000); insert into loan values(53487618722,2345,870000);* -- Borrower *insert into borrower values(8, 18273497653); insert into borrower values(15, 05674321567); insert into borrower values(12, 62348762342); insert into borrower values(16, 53487618722); insert into borrower values(18, 76528612384); insert into borrower values(2, 23487699172); insert into borrower values(19, 82539451438); insert into borrower values(3, 23475628222); insert into borrower values(22, 09862343172); insert into borrower values(7, 67523849211); insert into borrower values(21, 32487691465); insert into borrower values(11, 76342625333); insert into borrower values(20, 02934699255); insert into borrower values(14, 92734755222); insert into borrower values(5, 10192357333); insert into borrower values(10, 76342625343); insert into borrower values(9, 79263994636); insert into borrower values(1, 23487652982); insert into borrower values(13, 92348769444); insert into borrower values(6, 12012404280); insert into borrower values(17, 23763499521); insert into borrower values(4, 12359612875);* -- Depositor *insert into depositor values(8,112404863); insert into depositor values(15,1012404500); insert into depositor values(12,1012504576); insert into depositor values(16,112504526); insert into depositor values(18,112404017); insert into depositor values(2,1212305468); insert into depositor values(19,112404050); insert into depositor values(3,1012404586 ); insert into depositor values(22,112404117); insert into depositor values(7,1612103586); insert into depositor values(13,1112403497); insert into depositor values(21,1012404001); insert into depositor values(11,1212104580); insert into depositor values(20,2015404080); insert into depositor values(14,112604080); insert into depositor values(5,1112404586); insert into depositor values(10,1012404526); insert into depositor values(9,1012404562); insert into depositor values(1,1012305478); insert into depositor values(6,1012305481); insert into depositor values(17,112404217); insert into depositor values(4,1012104586);* |

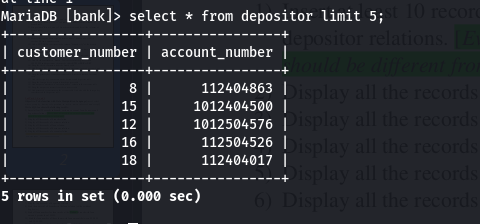
     

## 23) Display all the records

|  |
| --- |
| select \* from branch;  select \* from customer;  select \* from loan;  select \* from borrower;  select \* from account;  select \* from depositor; |

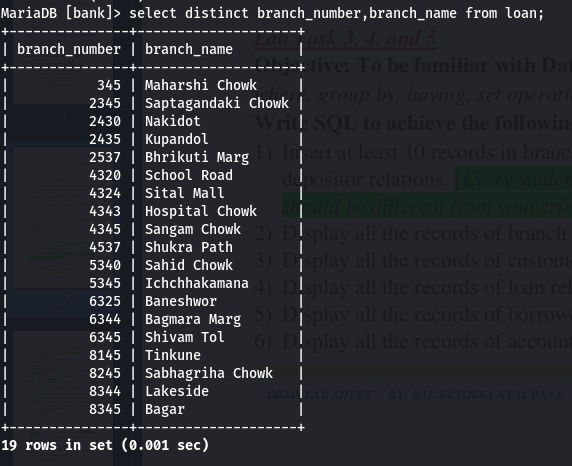
## 24) Display top 5 records of depositor relation

|  |
| --- |
| select \* from depositor limit 5; |



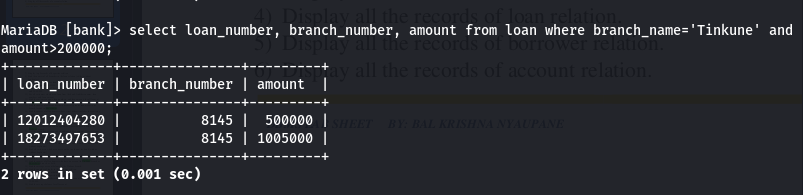
**25) Find the names of all branches in the loan relation (with or without duplicates).**

|  |
| --- |
| *select branch\_number,branch\_name from loan; select distinct branch\_number,branch\_name from loan;* |

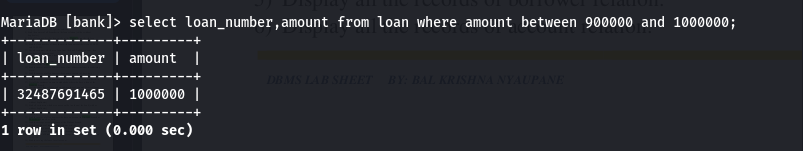
## 26) Find all loan numbers for loans made at the Tinkune branch with loan amounts greater than 200000.

|  |
| --- |
| select loan\_number, l.branch\_number, b.branch\_name from loan as l,branch as b where l.branch\_number=b.branch\_number and b.branch\_name='Tinkune' and amount>200000; |



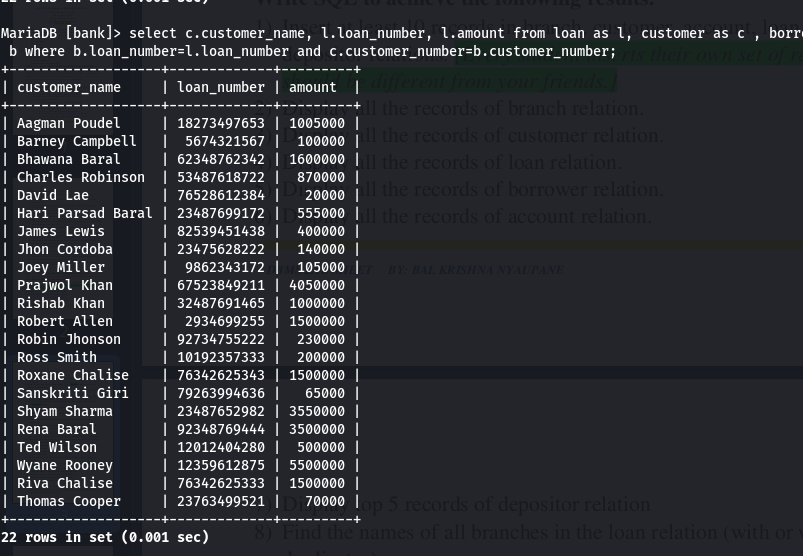
## 27) Find the loan number of those loans with loan amounts between 9,00,000 and 10,00,000.

|  |
| --- |
| select loan\_number,amount from loan where amount between 900000 and 1000000; |



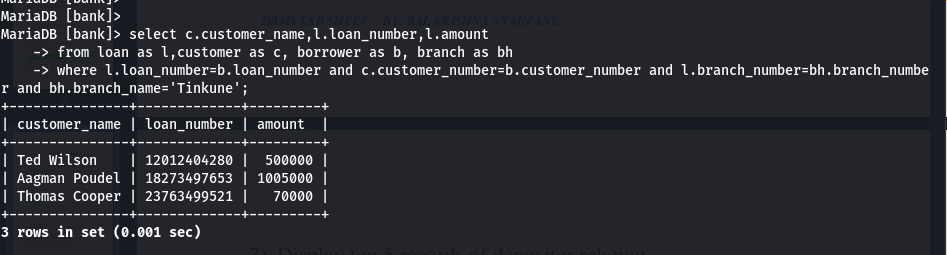
## 28) For all customers who have a loan from the bank, find their names, loan numbers, and loan amount.

|  |
| --- |
| select c.customer\_name, l.loan\_number, l.amount from loan as l, customer as c , borrower as b where b.loan\_number=l.loan\_number and c.customer\_number=b.customer\_number; |



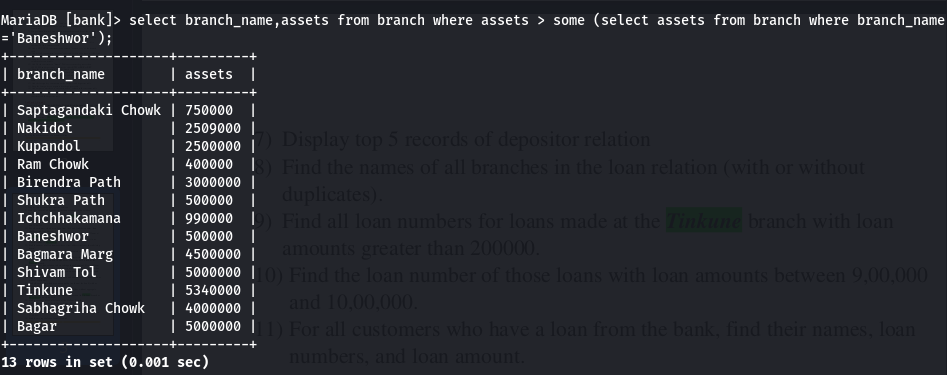
## 29) Find the customer names, loan numbers, and loan amounts for all loans at the Tinkune branch.

|  |
| --- |
| select c.customer\_name,l.loan\_number,l.amount  from loan as l,customer as c, borrower as b, branch as bh where l.loan\_number=b.loan\_number and c.customer\_number=b.customer\_number and l.branch\_number=bh.branch\_number and bh.branch\_name='Tinkune'; |



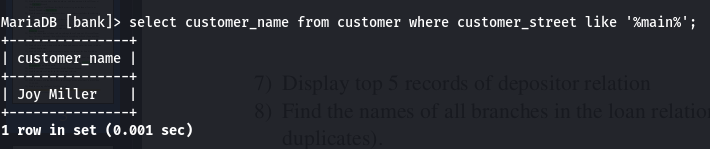
## 30) Find the names of all branches that have assets greater than at least one branch located in Baneshwor.

|  |
| --- |
| select branch\_name,assets from branch where assets > some (select assets from branch where branch\_name='Baneshwor'); |



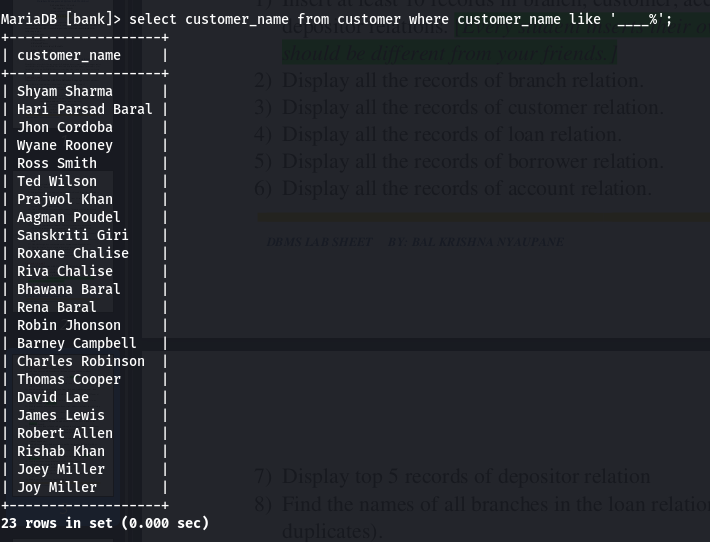
## 31) Find the names of all customers whose street address includes the substring ‘main’.

|  |
| --- |
| select customer\_name from customer where customer\_street like '%main%'; |



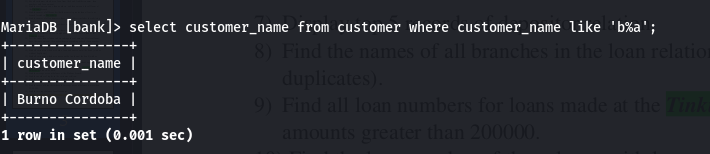
## 32) Find the names of all customers whose name contains at least four characters.

|  |
| --- |
| select customer\_name from customer where customer\_name like '\_\_\_\_%'; |



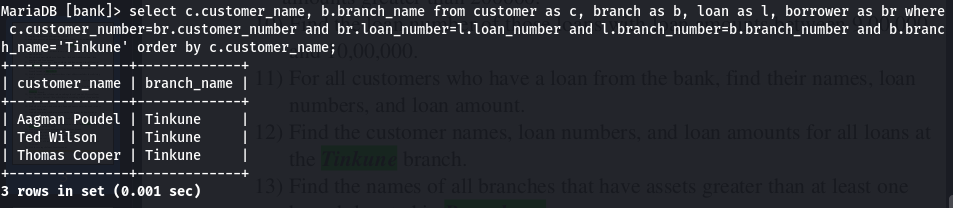
## 33) Find the names of all customers whose name start with ‘b’ and end with ‘a’.

|  |
| --- |
| select customer\_name from customer where customer\_name like 'b%a'; |



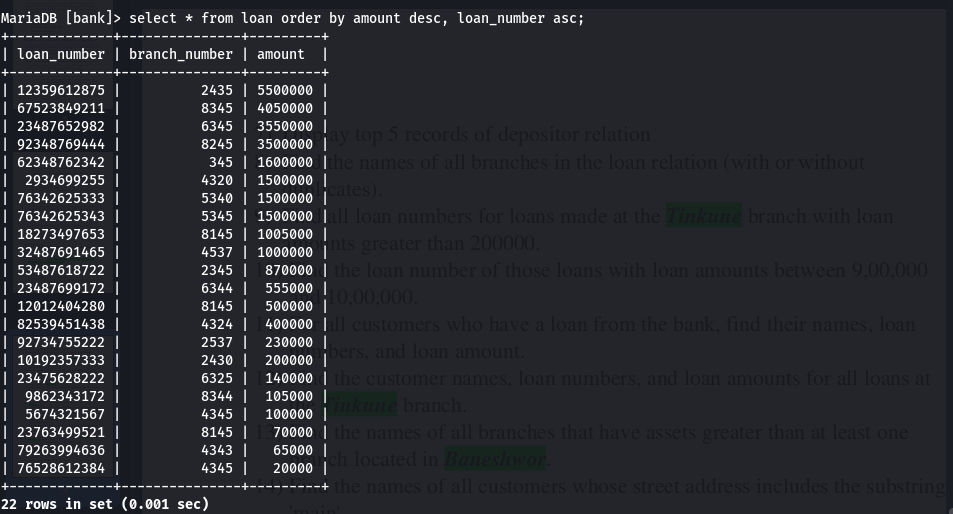
## 34) List all the customer’s name in alphabetic order who have a loan at the Tinkune branch.

|  |
| --- |
| *select c.customer\_name, b.branch\_name from customer as c, branch as b, loan as l, borrower as br where c.customer\_number=br.customer\_number and br.loan\_number=l.loan\_number and l.branch\_number=b.branch\_number and b.branch\_name='Tinkune' order by c.customer\_name;* |



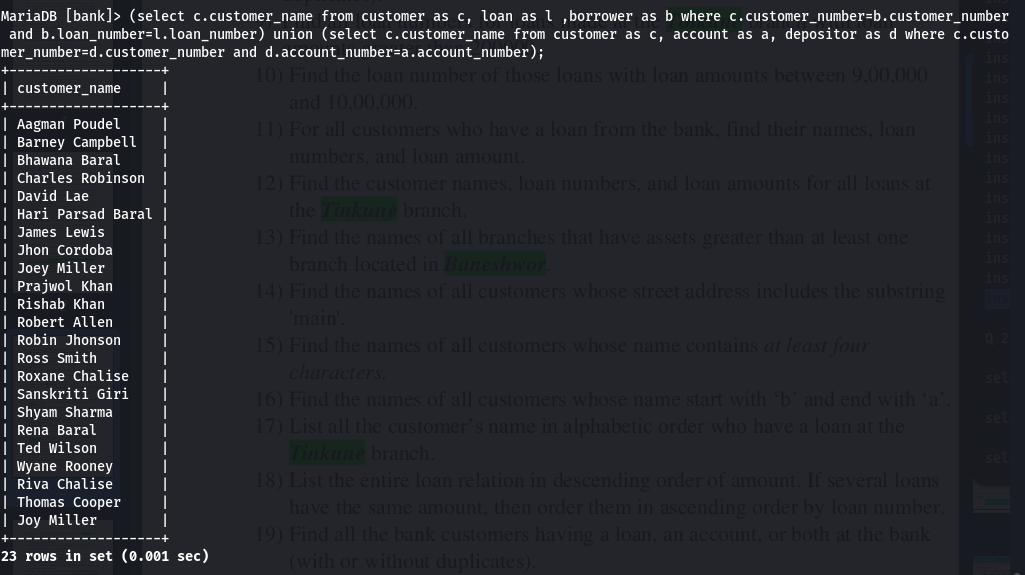
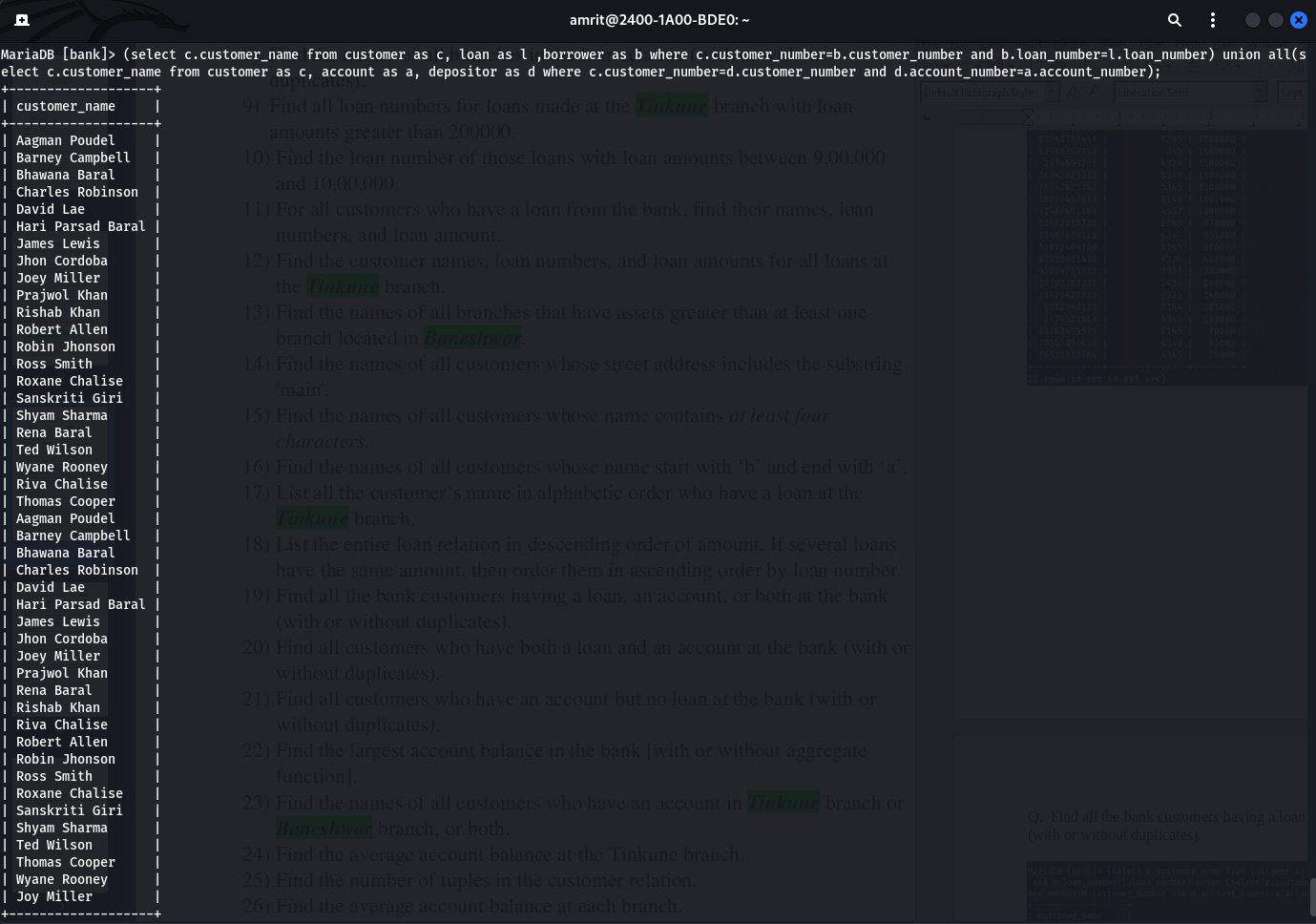
## 35) List the entire loan relation in descending order of amount. If several loans have the same amount, then order them in ascending order by loan number.

|  |
| --- |
| select \* from loan order by amount desc, loan\_number asc; |



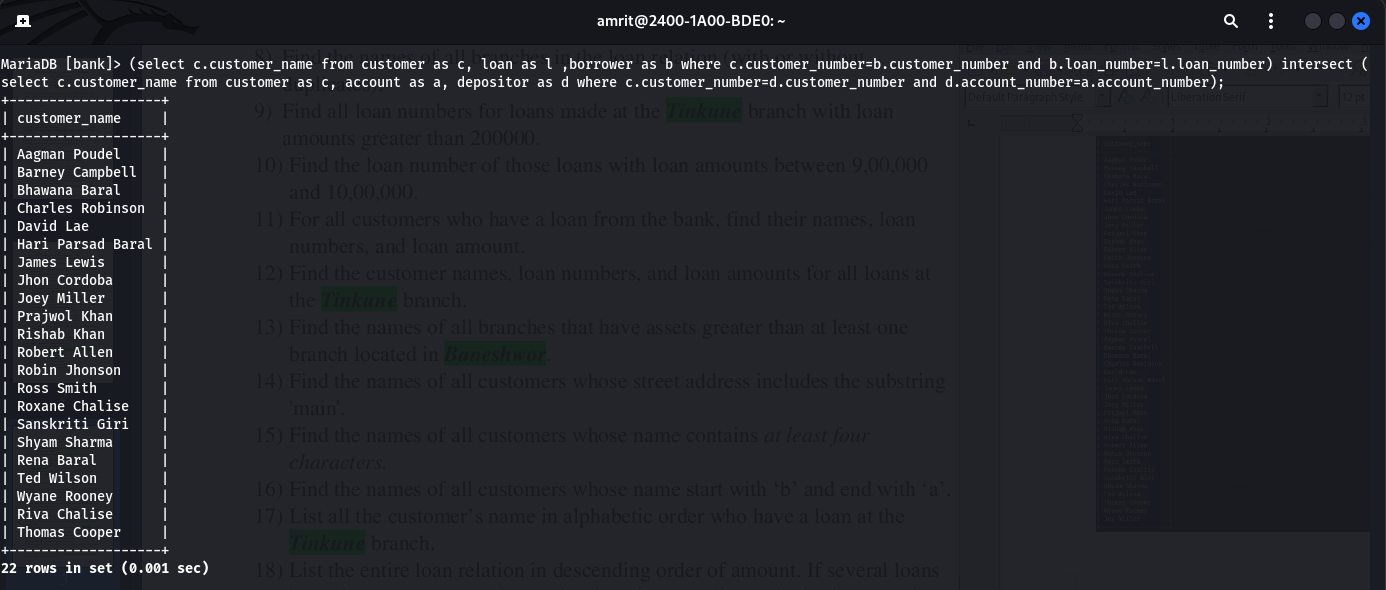
## 36) Find all the bank customers having a loan, an account, or both at the bank (with or without duplicates).

|  |
| --- |
| -- without duplicate (select c.customer\_name from customer as c, loan as l ,borrower as b where c.customer\_number=b.customer\_number and b.loan\_number=l.loan\_number)union(select c.customer\_name from customer as c, account as a, depositor as d where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number)  -- with duplicate (select c.customer\_name from customer as c, loan as l ,borrower as b where c.customer\_number=b.customer\_number and b.loan\_number=l.loan\_number)union all(select c.customer\_name from customer as c, account as a, depositor as d where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number) |

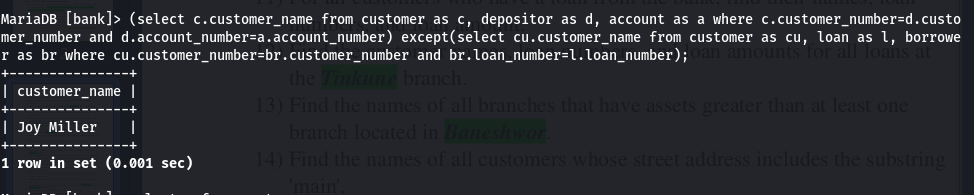
## 37) Find all customers who have both a loan and an account at the bank (with or without duplicates).

|  |
| --- |
| -- wihtout duplicate (select c.customer\_name from customer as c, loan as l ,borrower as b where c.customer\_number=b.customer\_number and b.loan\_number=l.loan\_number) intersect (select c.customer\_name from customer as c, account as a, depositor as d where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number);  -- with duplicate (select c.customer\_name from customer as c, loan as l ,borrower as b where c.customer\_number=b.customer\_number and b.loan\_number=l.loan\_number) intersect all(select c.customer\_name from customer as c, account as a, depositor as d where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number); |



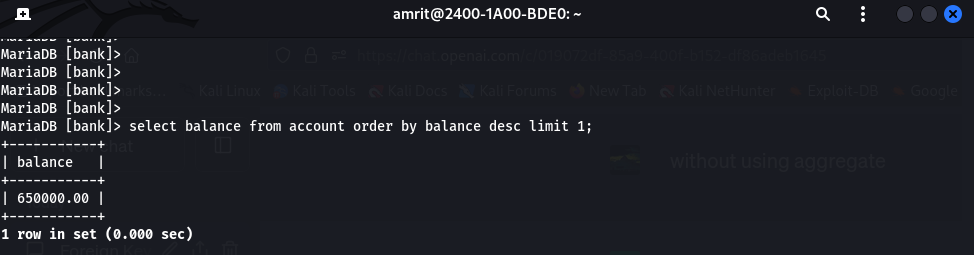
## 38) Find all customers who have an account but no loan at the bank (with or without duplicates).

|  |
| --- |
| -- Without duplicate (select c.customer\_name from customer as c, depositor as d, account as a where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number) except(select cu.customer\_name from customer as cu, loan as l, borrower as br where cu.customer\_number=br.customer\_number and br.loan\_number=l.loan\_number);  -- with duplicate (select c.customer\_name from customer as c, depositor as d, account as a where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number) except all(select cu.customer\_name from customer as cu, loan as l, borrower as br where cu.customer\_number=br.customer\_number and br.loan\_number=l.loan\_number); |



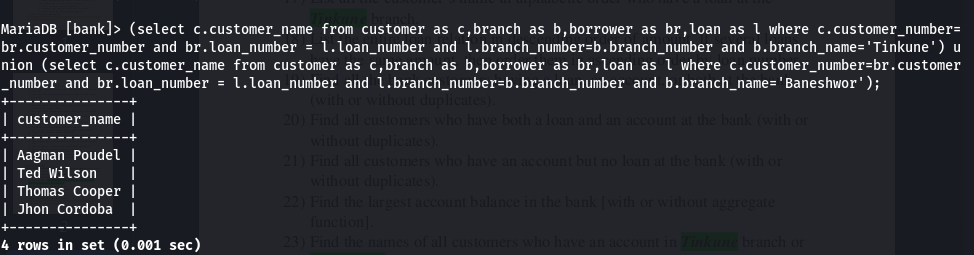
## 39) Find the largest account balance in the bank [with or without aggregate function].

|  |
| --- |
| -- without aggregate select balance from account order by desc limit 1; -- with aggregate select max(balance) from account; |

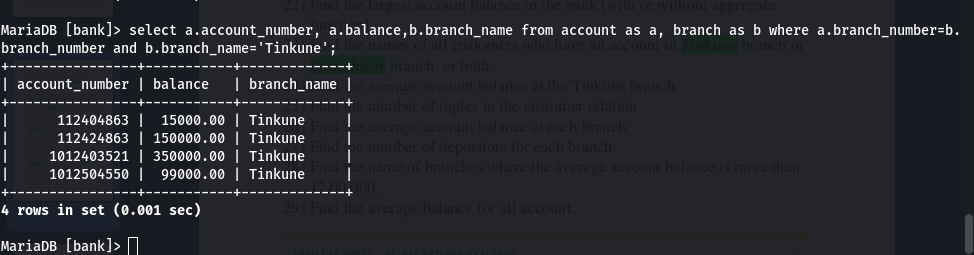
## 40) Find the names of all customers who have an account in Tinkune branch or Baneshwor branch, or both.

|  |
| --- |
| (select c.customer\_name from customer as c,branch as b,borrower as br,loan as l where c.customer\_number=br.customer\_number and br.loan\_number = l.loan\_number and l.branch\_number=b.branch\_number and b.branch\_name='Tinkune')  union  (select c.customer\_name from customer as c,branch as b,borrower as br,loan as l where c.customer\_number=br.customer\_number and br.loan\_number = l.loan\_number and l.branch\_number=b.branch\_number and b.branch\_name='Baneshwor'); |



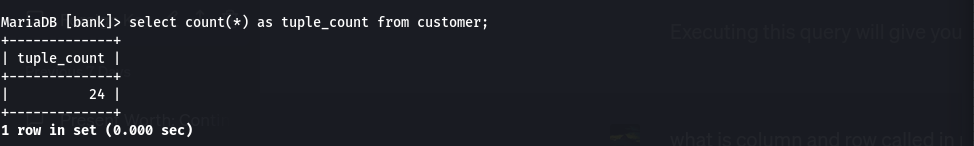
## 41) Find the average account balance at the Tinkune branch.

|  |
| --- |
| select a.account\_number, a.balance,b.branch\_name from account as a, branch as b where a.branch\_number=b.branch\_number and b.branch\_name='Tinkune'; |



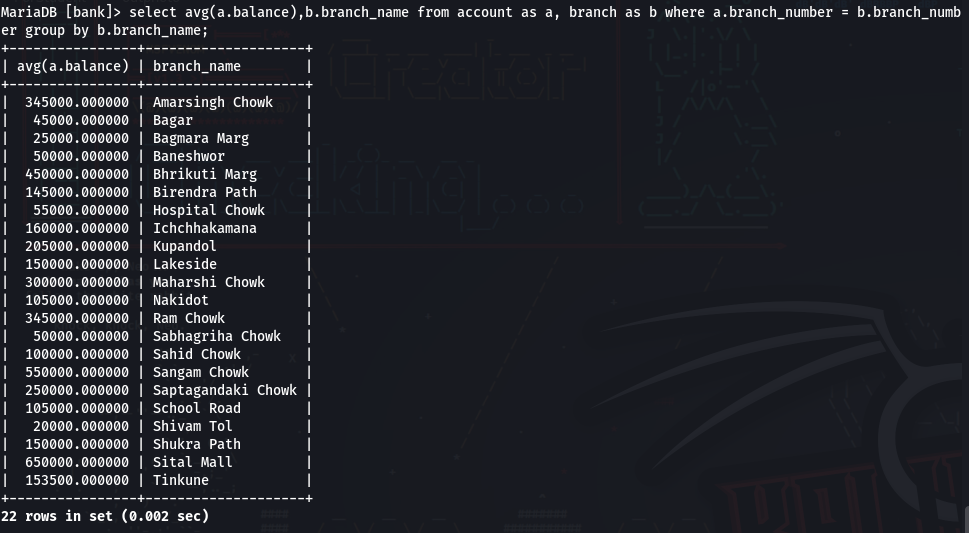
## 42) Find the number of tuples in the customer relation.

|  |
| --- |
| select count(\*) from customer; |



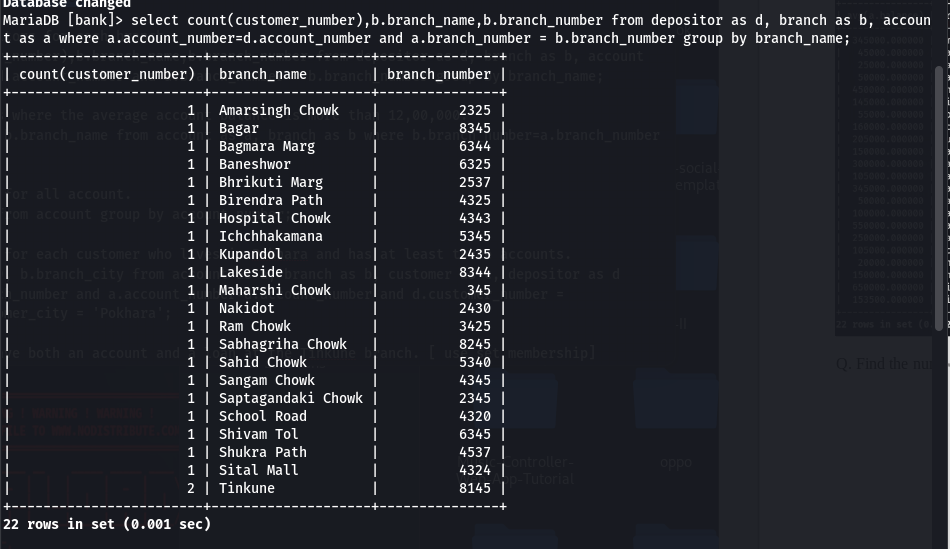
## 43) Find the average account balance at each branch.

|  |
| --- |
| select avg(a.balance),b.branch\_name from account as a, branch as b where a.branch\_number = b.branch\_number group by b.branch\_name; |



## 44) Find the number of depositors for each branch.

|  |
| --- |
| select count(customer\_number),b.branch\_name,b.branch\_number from depositor as d, branch as b, account as a where a.account\_number=d.account\_number and a.branch\_number = b.branch\_number group by branch\_name; |



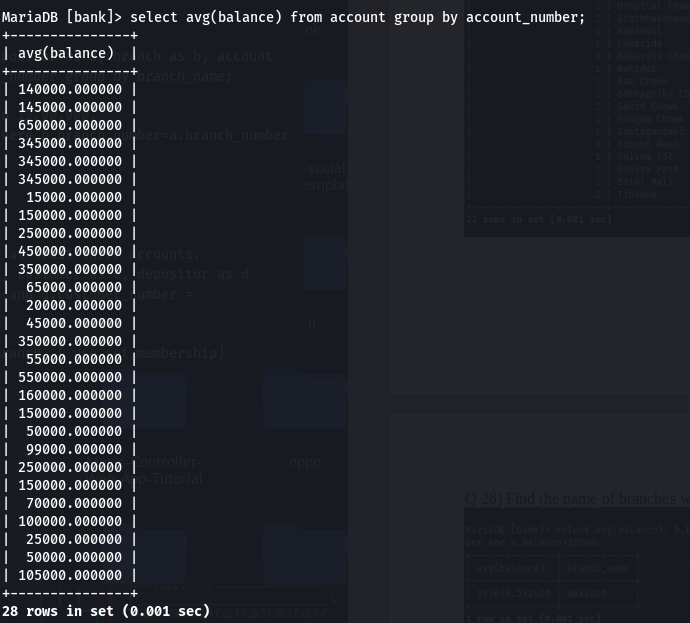
## 45) Find the name of branches where the average account balance is more than 12,00,000.

|  |
| --- |
| select avg(balance), b.branch\_name from account as a, branch as b where b.branch\_number=a.branch\_number and a.balance>1200000; |



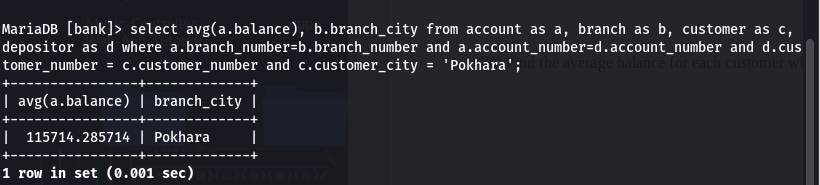
## 46) Find the average balance for all account.

|  |
| --- |
| select avg(balance) from account group by account\_number; |



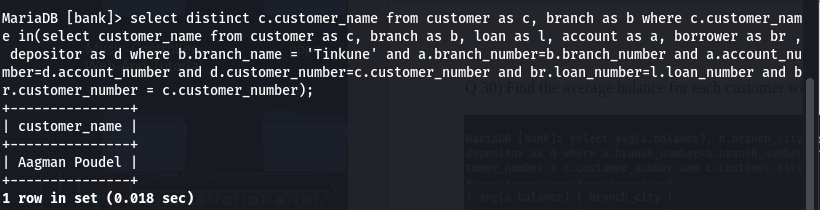
## 47) Find the average balance for each customer who lives in Pokhara and has at least three accounts.

|  |
| --- |
| select avg(a.balance), b.branch\_city from account as a, branch as b, customer as c, depositor as d where a.branch\_number=b.branch\_number and a.account\_number=d.account\_number and d.customer\_number = c.customer\_number and c.customer\_city = 'Pokhara'; |



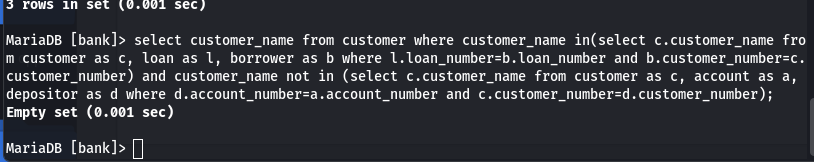
## 48) Find all customers who have both an account and a loan at the Tinkune branch. [ use set membership]

|  |
| --- |
| *select distinct c.customer\_name from customer as c, branch as b where c.customer\_name in(select customer\_name from customer as c, branch as b, loan as l, account as a, borrower as br , depositor as d where b.branch\_name = 'Tinkune' and a.branch\_number=b.branch\_number and a.account\_number=d.account\_number and d.customer\_number=c.customer\_number and br.loan\_number=l.loan\_number and br.customer\_number = c.customer\_number);* |



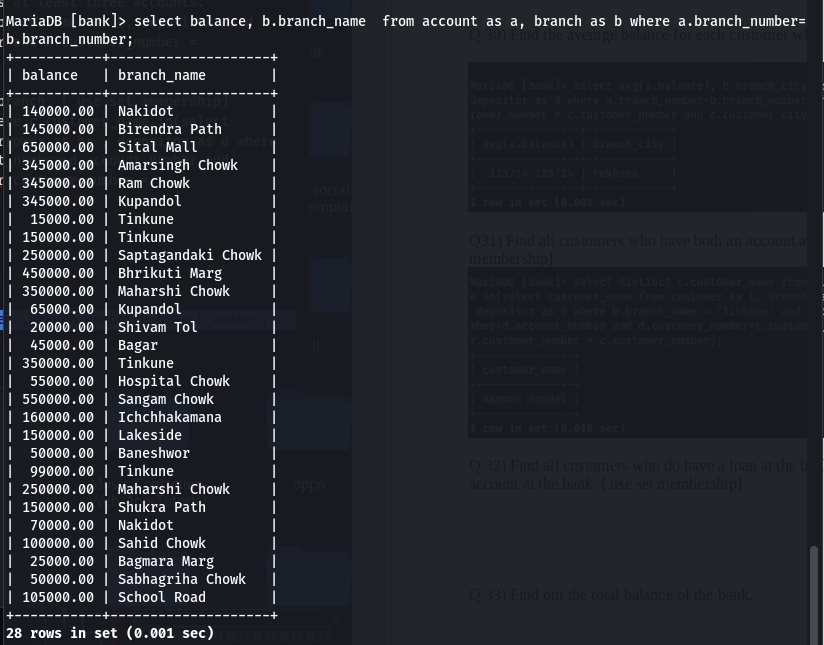
## 49) Find all customers who do have a loan at the bank, but do not have an account at the bank. [ use set membership]

|  |
| --- |
| *select distinct c.customer\_name from customer as c, branch as b where c.customer\_name in(select customer\_name from customer as c, loan as l, account as a, borrower as br, depositor as d where );* |



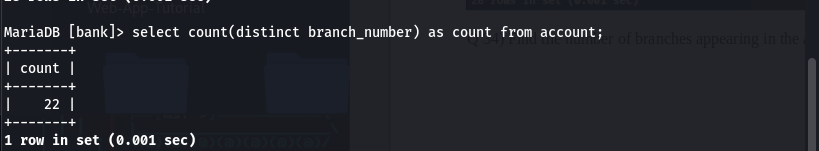
## 50) Find out the total balance of the bank.

|  |
| --- |
| select balance, b.branch\_name from account as a, branch as b where a.branch\_number=b.branch\_number; |



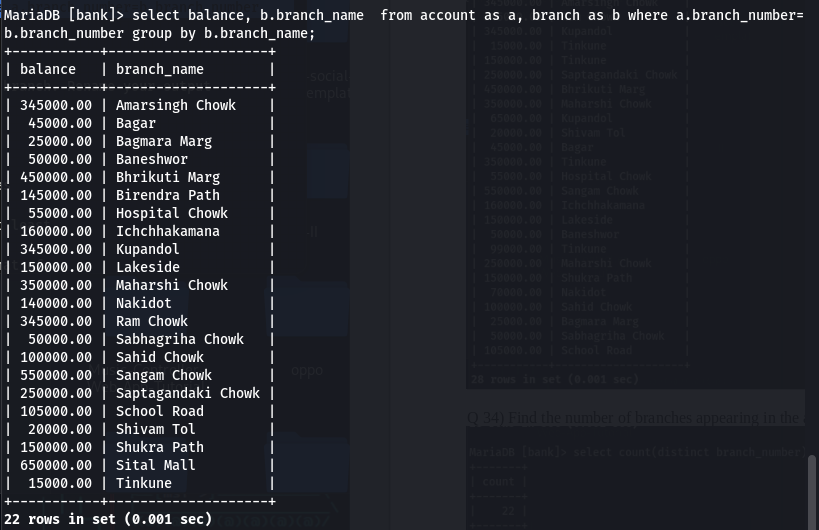
## 51) Find the number of branches appearing in the account relation.

|  |
| --- |
| select count(distinct branch\_number) as count from account; |



## 52) Find the total balance of each branch of the bank.

|  |
| --- |
| *select balance, b.branch\_name from account as a, branch as b where a.branch\_number=b.branch\_number group by b.branch\_name;* |



## 53) Find the maximum balance at each branch and sum of the balance of each branch. Rename your output attributes.

|  |
| --- |
| *select max(a.balance) as max\_balance, sum(a.balance),a.branch\_number as sum\_balance from account as a group by a.branch\_number;* |



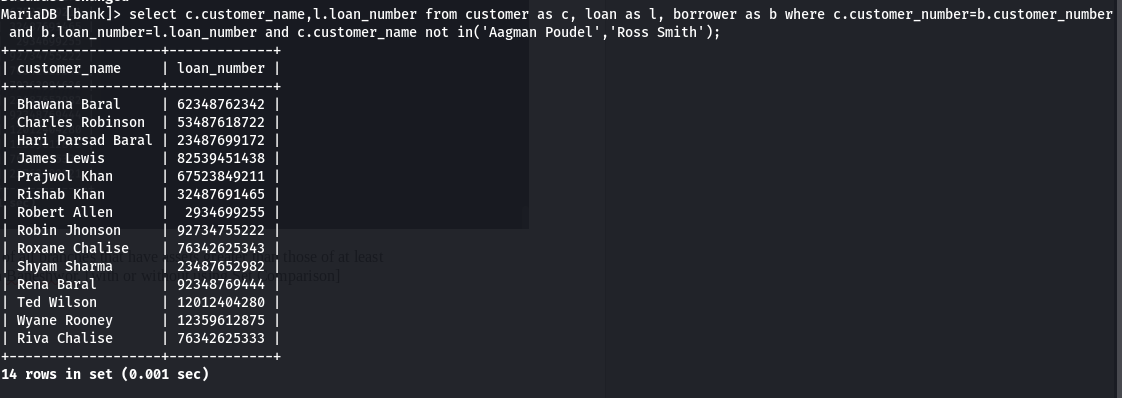
## 54) List the names of customers who have a loan at the bank, and whose names are neither Shyam nor Hari.

|  |
| --- |
| select c.customer\_name,l.loan\_number from customer as c, loan as l, borrower as b where c.customer\_number=b.customer\_number and b.loan\_number=l.loan\_number and c.customer\_name not in('Aagman Poudel','Ross Smith'); |



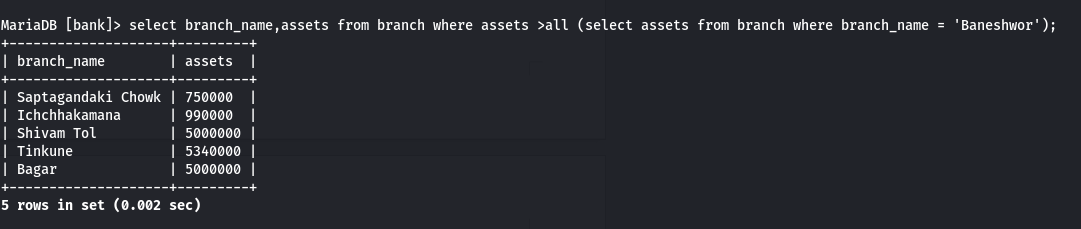
## 55) Find the name of all branches that have assets greater than those of at least one branch located in Baneshwor. [with or without using Set Comparison]

|  |
| --- |
| select branch\_name,assets from branch where assets >some (select assets from branch where branch\_name = 'Baneshwor'); |



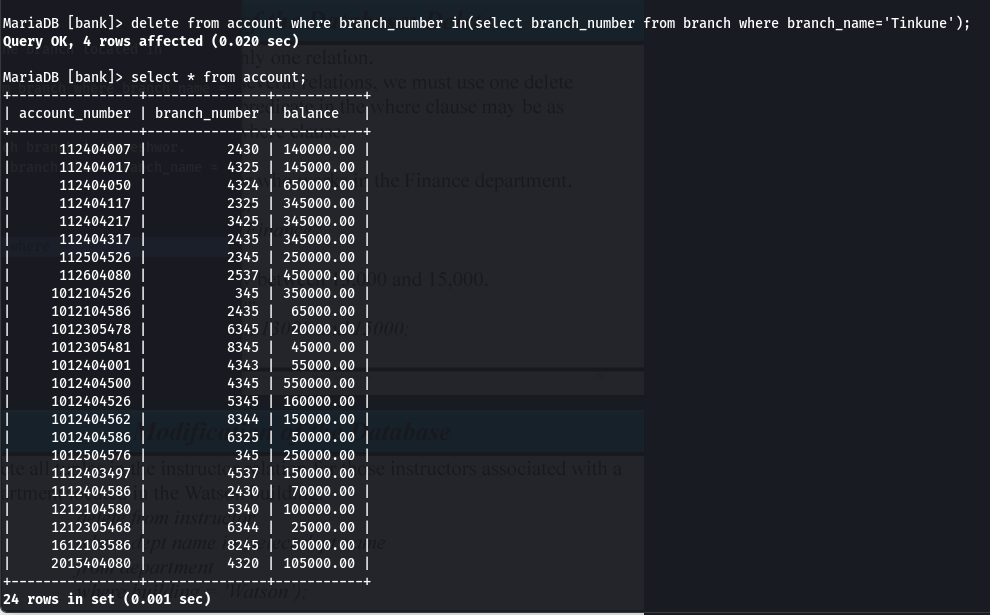
## 56) Find the names of all branches that have an asset value greater than that of each branch in Baneshwor.

|  |
| --- |
| select branch\_name,assets from branch where assets >all (select assets from branch where branch\_name = 'Baneshwor'); |



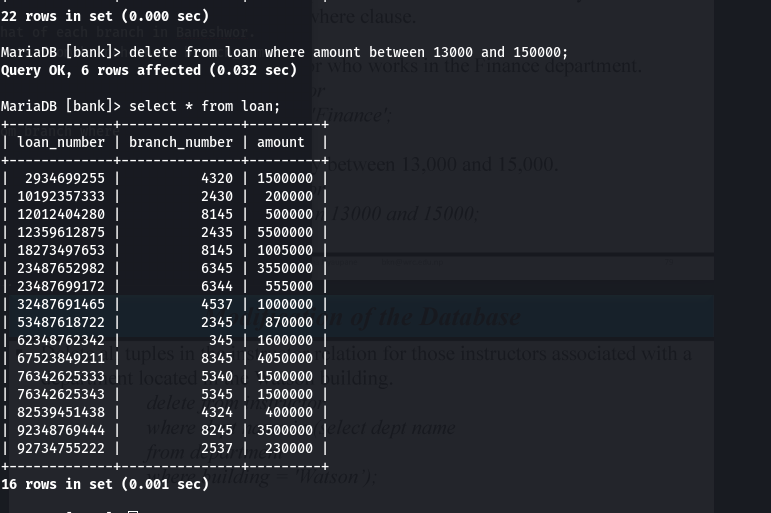
## 57) Delete all account tuples in the Tinkune branch.

|  |
| --- |
| delete from account where branch\_number in(select branch\_number from branch where branch\_name='Tinkune'); |



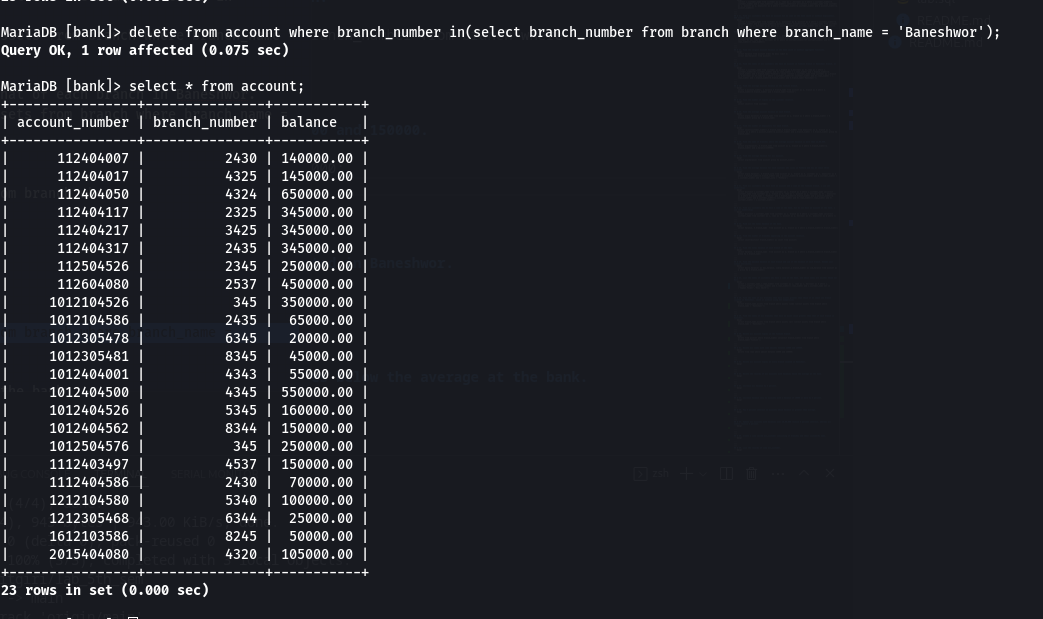
## 58) Delete all loans with loan amounts between 13000 and 150000.

|  |
| --- |
| delete from loan where amount between 13000 and 150000; |



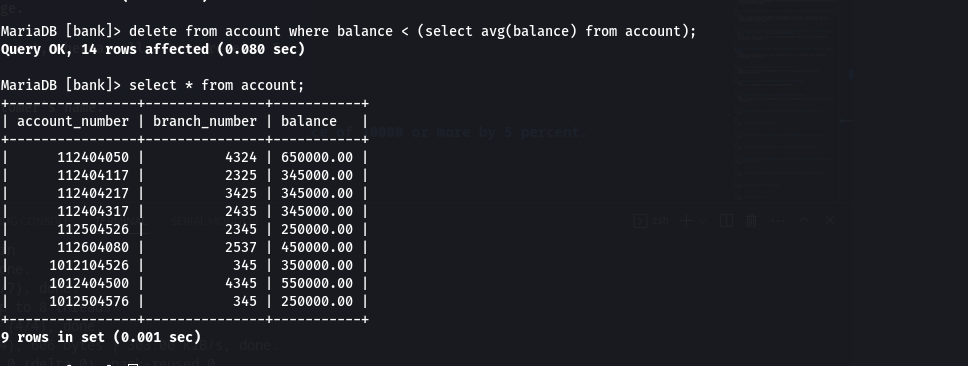
## 59) Delete all account tuples at every branch located in Baneshwor.

|  |
| --- |
| delete from account where branch\_number in(select branch\_number from branch where branch\_name = 'Baneshwor'); |



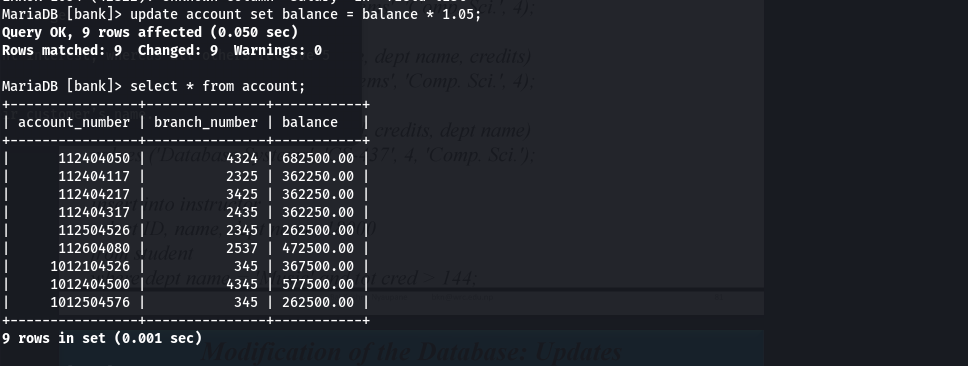
## 60) Delete the records of all accounts with balances below the average at the bank.

|  |
| --- |
| delete from account where balance < (select avg(balance) from account); |



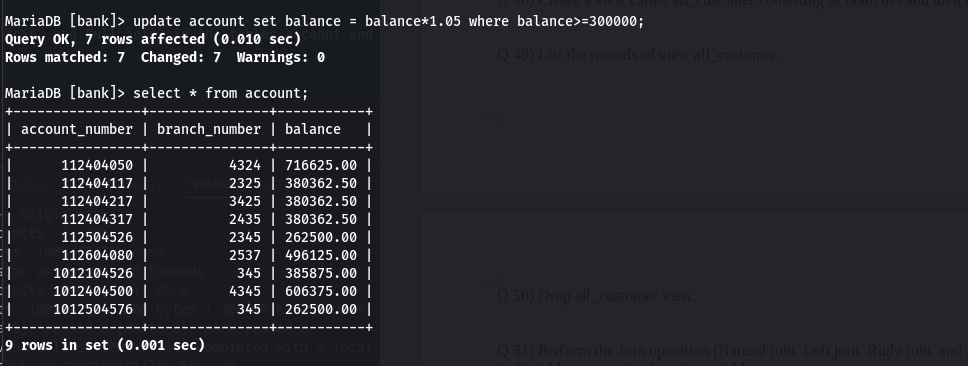
## 61) Increase all balance by 5 percent.

|  |
| --- |
| update account set balance = balance \* 1.05; |



## 62) Increase balance only to accounts with a balance of 10000 or more by 5 percent.

|  |
| --- |
| update account set balance = balance\*1.05 where balance>=300000; |



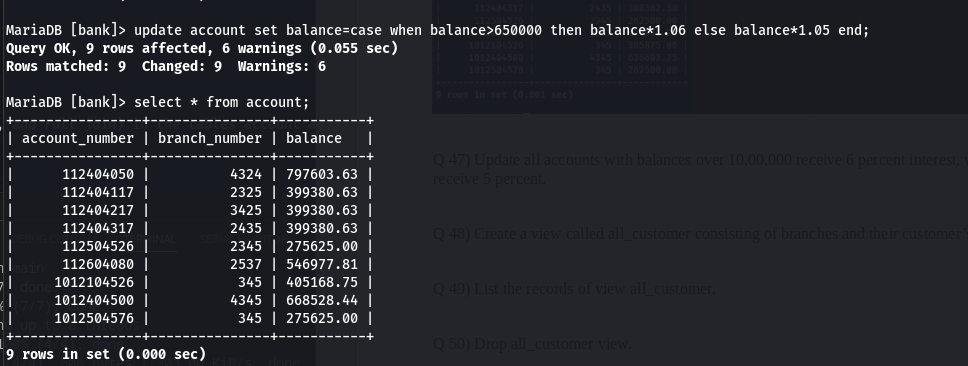
## 63) Pay 5 percent interest on accounts whose balance is greater than average.

|  |
| --- |
| update account set balance = balance\*1.05 where balance > (select avg(balance) from account); |



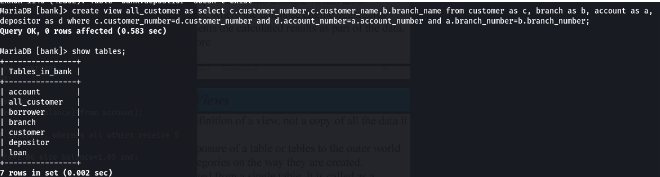
## 64) Update all accounts with balances over 10,00,000 receive 6 percent interest, whereas all others receive 5 percent.

|  |
| --- |
| update account set balance=case when balance>650000 then balance\*1.06 else balance\*1.05 end; |



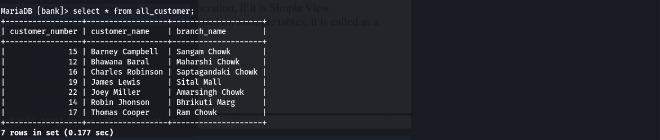
## 65) Create a view called all\_customer consisting of branches and their customer’s name.

|  |
| --- |
| create view all\_customer as select c.customer\_number,c.customer\_name,b.branch\_name from customer as c, branch as b, account as a, depositor as d where c.customer\_number=d.customer\_number and d.account\_number=a.account\_number and a.branch\_number=b.branch\_number; |



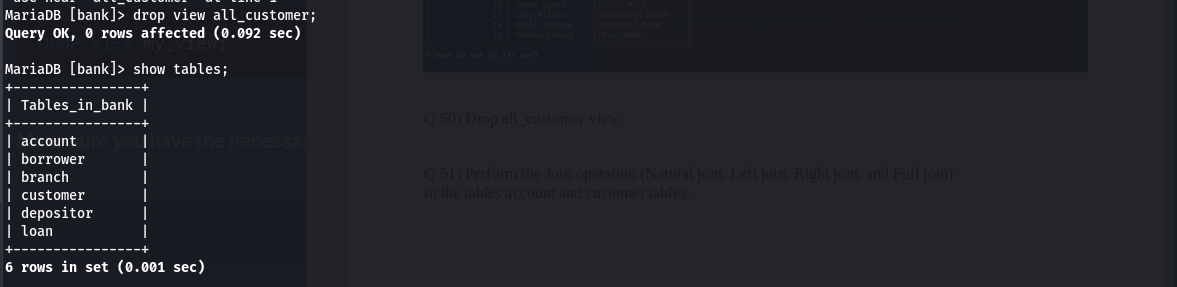
## 66) List the records of view all\_customer.

|  |
| --- |
| *select \* from all\_customer;* |



## 67) Drop all\_customer view.

|  |
| --- |
| drop view all\_customer; |



## 68) Perform the Join operation (Natural join, Left join, Right join, and Full join) in the tables account and customer tables.

## 

|  |
| --- |
| -- Natural Join select \* from account natural join depositor natural join customer;  -- left join select \* from account left join depositor on account.account\_number=depositor.account\_number left join customer on depositor.customer\_number=customer.customer\_number;  -- Right Join select \* from account right join depositor on account.account\_number=depositor.account\_number right join customer on depositor.customer\_number=customer.customer\_number;  -- Full Join select \* from account full join depositor on account.account\_number=depositor.account\_number full join customer on depositor.customer\_number=customer.customer\_number;  -- "One that doesnot support full join" (select \* from account left join depositor on account.account\_number=depositor.account\_number left join customer on depositor.customer\_number=customer.customer\_number) union (select \* from account right join depositor on account.account\_number=depositor.account\_number right join customer on depositor.customer\_number=customer.customer\_number); |

